

This Page Is Inserted by IFW Operations
and is not a part of the Official Record

BEST AVAILABLE IMAGES

Defective images within this document are accurate representations of the original documents submitted by the applicant.

Defects in the images may include (but are not limited to):

- BLACK BORDERS
- TEXT CUT OFF AT TOP, BOTTOM OR SIDES
- FADED TEXT
- ILLEGIBLE TEXT
- SKEWED/SLANTED IMAGES
- COLORED PHOTOS
- BLACK OR VERY BLACK AND WHITE DARK PHOTOS
- GRAY SCALE DOCUMENTS

IMAGES ARE BEST AVAILABLE COPY.

**As rescanning documents *will not* correct images,
please do not report the images to the
Image Problem Mailbox.**

This Page Blank (uspto)



The
Patent
Office



INVESTOR IN PEOPLE

GB 93/13229

**PRIORITY
DOCUMENT**

SUBMITTED OR TRANSMITTED IN
COMPLIANCE WITH RULE 17.1(a) OR (b)

The Patent Office
Concept House
Cardiff Road
Newport
South Wales
NP10 8QQ

REC'D 23 NOV 1999

WIPO PCT

I, the undersigned, being an officer duly authorised in accordance with Section 74(1) and (4) of the Deregulation & Contracting Out Act 1994, to sign and issue certificates on behalf of the Comptroller-General, hereby certify that annexed hereto is a true copy of the documents as originally filed in connection with the patent application identified therein.

In accordance with the Patents (Companies Re-registration) Rules 1982, if a company named in this certificate and any accompanying documents has re-registered under the Companies Act 1980 with the same name as that with which it was registered immediately before re-registration save for the substitution as, or inclusion as, the last part of the name of the words "public limited company" or their equivalents in Welsh, references to the name of the company in this certificate and any accompanying documents shall be treated as references to the name with which it is so re-registered.

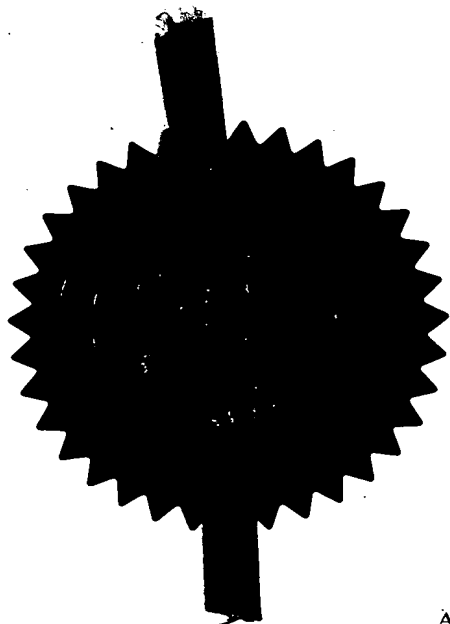
In accordance with the rules, the words "public limited company" may be replaced by p.l.c., plc, P.L.C. or PLC.

Re-registration under the Companies Act does not constitute a new legal entity but merely subjects the company to certain additional company law rules.

Signed

Dated

10 NOV 1999



This Page Blank (uspto)

Patents Form 1/77
Patents Act 1977
(Rule 16)




30SEP98 E393470-1 D02917
P01/7700 25.00 - 9821139.4

Request for grant of a patent

The Patent Office
Cardiff Road
Newport
Gwent NP9 1RH

1. Your reference
5279301/KB
2. Patent Appl.
9821139.4 29 SEP 1998
3. Full name, address and postcode of the or of each applicant (*underline all surnames*)

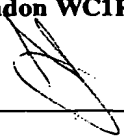
University College London
Gower Street
London WC1E 6BT

798652002 

Patents ADP number (*if known*)

If the applicant is a corporate body, give the country/state of its incorporation Country: ENGLAND
4. Title of the invention
ENERGY PROPAGATION MODELLING
5. Name of agent Beresford & Co

"Address for Service" in the United Kingdom to which all correspondence should be sent 2/5 Warwick Court
High Holborn
London WC1R 5DJ

Patents ADP number 1826001 
6. Priority details

Country Priority application number Date of filing

Patents Form 1/77

7. If this application is divided or otherwise derived from an earlier UK application give details

Number of earlier of application

Date of filing

8. Is a statement of inventorship and or right to grant of a patent required in support of this request?

YES

9. Enter the number of sheets for any of the following items you are filing with this form.

Continuation sheets of this form

Description

129

Claim(s)

9

Abstract

1

Drawing(s)

32

10. If you are also filing any of the following, state how many against each item.

Priority documents

Translations of priority documents

Statement of inventorship and
right to grant of a patent (*Patents form 7/77*)

Request for preliminary examination
and search (*Patents Form 9/77*)

1

Request for Substantive Examination
(*Patents Form 10/77*)

Any other documents
(*please specify*)

11. I/We request the grant of a patent on the basis of this application

Signature


BERESFORD & Co

Date

29 September 1998

12. Name and daytime telephone number of
person to contact in the United Kingdom

DAVID SPROSTON

Tel:0171-831-2290

DUPLICATE

1

ENERGY PROPAGATION MODELLING

The present invention is concerned with the modelling of the propagation of energy. In particular, the present invention is concerned with the modelling of propagation
5 which can be represented by ray paths, and the manner in which energy passes along ray paths while interacting with objects in a modelled scene. The present invention has applications in the modelling of light in a scene, to provide a facility for production of near photo-
10 realistic images of a scene or to provide for measurement of light distribution within a scene.

For a considerable period of time, it has been accepted that two main qualities of computer graphics exist,
15 namely photo-realism and real-time dynamic imagery. Due to constraints of computer capacity and processing speed, it has been recognised that one of these qualities has to be sacrificed at least to some extent in order to improve the other. In many computer games, dynamic
20 imagery is of real importance, and so real-time production of dynamic images is implemented to the detriment of photo-realism. On the other hand, many applications do not require dynamic changes to an image in real-time, whereas photo-realism is of real
25 importance. However, even in a photo-realistic model, it is preferable that a user can change the viewpoint of a computer modelled scene, with limited re-calculation.

If this can be carried out without significant recalculation, the technique is considered to be "view-independent". View-independence in this sense means that data pertaining to the model of the scene does not vary
5 on variation of the point of views of the scene.

In order to model a scene in a photo-realistic manner, the true behaviour of light incident on objects in the scene must be approximated. In particular, the
10 reflection and transmission of light must be modelled so as to approximate to the behaviour of light in real life. In that way, the observer of the modelled scene will reach the correct conclusions concerning the nature and appearance of objects within the scene.

15

Light within a scene has in the past been modelled by several techniques, most of which can be categorised as either "ray tracing" techniques or "radiosity" techniques.

20

Ray tracing is described in detail in "An Improved Illumination Model for Shaded Display" by T. Whitted, published in COMM.AMC 23(6), 343-349, June 1980. Ray tracing assumes that the observer is viewing an image
25 projected through a pin-hole camera, on to an image plane. The image is of a scene illuminated by point light sources. For ease of calculation, the medium

through which light is transmitted is usually assumed to be non-participating, but light attenuation effects can be modelled as described in "Computer Graphics, Principles and Practice" by Foley, van Dam, Feiner, Hughes, 2nd ed. (hereinafter referred to as "Foley") pages 727 and 728.

Rays are traced from the pin-hole or centre of projection (COP) through the image plane and into the scene, bouncing from object to object using the laws of specular reflection (angle of reflection equals angle of incidence, incident and reflected rays are co-planar), until the ray exits the scene or it is established that further reflections will make negligible contribution to the image to be displayed on the computer screen because the intensity of further reflected rays is negligible. In practice, when considering a scene including transparent or translucent objects, transmission of light is considered in the same way as reflection, using well-known physical laws.

It should be noted at this point that the direction of ray tracing using the method described above is in the opposite direction to the actual direction of transmission of light. Ray tracing provides a method of modelling light at a predetermined viewpoint by firing rays from that point and computing reflections

encountered by those rays.

Ray tracing is entirely view-dependent. That means that if a different viewpoint is selected, the transmission of light towards the viewpoint from the scene must be entirely re-calculated. Moreover, it is not suitable for modelling diffuse reflection, where a particular incident ray results in a plurality of reflected ray directions, and a particular reflected ray can be the result of a plurality of incident rays. It is impossible to represent diffuse reflection realistically using ray tracing.

Several other modifications of ray tracing have been proposed, but with many drawbacks. In several cases, diffuse surfaces can be considered, alongside specular reflecting and transmitting surfaces. However, in those cases, diffusely reflecting surfaces are treated using discrete elements thereof, and subsequent shading to eliminate discontinuities between those discrete elements can result in specularly reflecting surfaces not being displayed as such.

Those methods include Ray Space Methods, as presented by Arvo & Kirk in Computer Graphics (SIGGRAPH) (1987) 21(4) pp 55-64, and by Muller & Winckler in "Lecture Notes in Computer Science (1992), 594, 124-147", Monte Carlo Path

Tracing, as proposed by Kajuya in Computer Graphics (SIGGRAPH) (1986) 20(4) pp 143-150, Photon Tracing, as introduced by Jensen & Christensen (1995) and refined by Shirley (1995) and Walter, Hubbard, Shirley and Greenberg (1997), and Discrete Ray Tracing as presented by Yager, Cohen and Kaufman (1992) (IEEE Computer Graphics & Applications 12(5), 14-28).

The second class of techniques, known as radiosity techniques, were introduced in the paper "Modelling the interaction of light between diffuse surfaces" by C M Goral et al, published in Computer Graphics (SIGGRAPH 84), 213-222. Radiosity techniques are suitable for scenes where all materials are ideal diffuse reflectors. However, radiosity is not suitable for consideration of specular reflection. Radiosity is defined as the rate at which light energy leaves a surface.

Almost all radiosity methods rely on a scene to be modelled being described in terms of polygons. Each polygon can be further sub-divided into small patches. The patches may in turn be sub-divided into surface elements. Each patch (or surface element as the case may be) is considered in turn, and the radiosity at that patch is calculated from the radiosity of patches from which light can be transmitted to the patch in question, taking into account their relative position and angular

orientations. It should be appreciated that radiosity techniques cannot be used for point light sources, since a light source must be capable of description in terms of polygons defining the geometry of the light source.

5

Once the radiosity at each patch has been calculated, Gouraud shading can be employed to produce a smooth image. Gouraud shading is described in Foley, pp 736 and 737. Since patches of finite area are considered instead
10 of points on a surface, radiosity will not result in accurate consideration of specular reflection. However, the lack of specular reflective effects means that the lighting of a model considered through radiosity techniques is independent of the position of the
15 viewpoint of the model. That is, recalculation of the whole image need not take place in the event that the position of the viewpoint changes. This is described as "view independence" in an alternative sense of the term.

20 As a result, a radiosity model is suitable to be viewed from a dynamic viewpoint in real time. This is known as "real-time walk-through". This is highly suitable for CAD applications such as for architectural models, but the loss of specular reflection results in a loss in
25 photo-realism. It should be appreciated that significant recalculation will be necessary at every change of view due to the need to determine visible surfaces and to

apply interpolative shading.

Finally, a light field rendering technique has been proposed, on the one-hand by Levoy and Hanrahan in a paper published in Computer Graphics (SIGGRAPH), Annual Conference Series (1996), 31-42, and on the other-hand by Gortler et al in Computer Graphics (SIGGRAPH), Annual Conference Series (1996), 43-52. This new method does not fall within the scope of either ray tracing or radiosity methods, but may make use of aspects of both.

In both disclosures, a light field, or lumigraph, is constructed of a pair of parallel planes, between which extend an ordered plurality of rays. Other light fields can be constructed using different arrangements of construction planes. The rays are arranged between discrete points on the planes, and can be described in terms of the points on the two planes between which they extend. A scene consisting of one or more objects can be rendered into the light field, by considering views of the scene at each point on the two planes. Those views can either be constructed by one of the techniques identified above, or by importing real views of a scene produced on digital cameras. Radiances along the rays from a particular point constructing the light field can be identified from the view from that point.

Once the radiances associated with rays from each point have been identified, views of the scene from a point within the light field can be constructed by considering the radiance of rays passing near that point and
5 interpolating an image from the radiance values obtained.

The technique is useful for rendering real images of real objects into the light field, so that images from positions other than the positions from which images were
10 obtained by a camera can also be seen. A virtual scene can be rendered into the light field, but it is necessary to produce a plurality of images of the scene, each from a different viewpoint, by a technique such as ray tracing or radiosity, before rendering into the light field can
15 be achieved. However, the light field technique described above has a significant disadvantage, in that views cannot be constructed from every position in the light field. If a scene consists of a plurality of objects, or at least one object which has at least one
20 concave surface, then a volume can be defined, between a surface bounding a minimum volume about the scene which includes no concave surfaces (the convex hull of the scene) and the actual surface(s) defining the object(s) of the scene. Several rays within this volume are
25 bounded at both ends by surfaces of objects in the scene. Therefore, radiances along those rays cannot be determined by considering real images at points on the

two planes. As a result, the volume is subject to discontinuities which prevent acceptable images being constructable within the volume.

5 It is an object of the present invention to provide an alternative technique for modelling energy propagation, such as the propagation of light through a scene. The invention provides simulation apparatus which defines discrete paths within an environment within which energy
10 propagation is to be modelled, and which locates points along those paths which denote interactions of objects within the environment with the paths.

One embodiment of the present invention provides a
15 technique which allows specular and diffuse reflection to be represented realistically. That embodiment further allows an image produced from a viewpoint of a three-dimensional scene to be changed in substantially constant time. That constant time is capable of being so small
20 that the image can be changed in real-time. The constant time is in principle not dependent on the complexity of the scene to be modulated.

The embodiment identified above allows an image to be
25 produced from a viewpoint anywhere within a volume bounding the scene to be modelled, without limitation to points outside of a particular volume.

Further features and advantages of the invention will become apparent from the following description of apparatus and a method in accordance with a preferred and
5 specific embodiment of the invention, with reference to the accompanying drawings in which:

Figure 1 is a schematic diagram of image processing apparatus in accordance with a specific embodiment of the
10 invention;

Figure 2A is a perspective view of a scene to be modelled by the image processing apparatus of Figure 1, including illustration of ray directions encountered at a first
15 viewpoint;

Figure 2B is a perspective view of the scene illustrated in Figure 2A, including illustration of ray directions encountered at a second viewpoint;
20

Figure 3A is a perspective view of a parallel sub-field in accordance with a specific embodiment of the present invention;

25 Figure 3B is a perspective view of a further parallel sub-field in accordance with a specific embodiment of the present invention;

Figure 3C is a perspective view of a still further parallel sub-field in accordance with a specific embodiment of the present invention;

5

Figure 4 is a perspective view of the scene illustrated in Figures 2A and 2B, including illustration of a selected plane within the scene to be modelled and a selected ray within that plane in accordance with the specific embodiment of the present invention;

10

Figure 5 is an elevation of the construction plane illustrated in Figure 4 in a direction normal to the construction plane, including a virtual eye in accordance with the specific embodiment of the present invention;

15

Figure 6 is a schematic diagram illustrating intersections of objects with the light ray illustrated in Figure 4, throughout its length;

20

Figure 7 is a schematic diagram showing an extract of the length of the light ray, between the side walls of the scene illustrated in Figure 4;

Figure 8 is a schematic diagram showing the structure of interval data in accordance with the specific embodiment of the present invention;

25

Figure 9 is a schematic diagram of a data structure for the interval data illustrated in Figure 8;

5 Figure 10 is a schematic diagram showing the structure of the image processing apparatus illustrated in Figure 1;

Figure 11 is a schematic diagram showing the structure
10 of the L F computation unit illustrated in Figure 10;

Figure 12 is a schematic diagram showing the internal structure of the viewer illustrated in Figure 10;

15 Figure 13 is a flow diagram illustrating the procedure performed by the image processing apparatus in use;

Figures 14 to 19 are flow diagrams illustrating sub-procedures called by the procedure illustrated in Figure
20 13;

Figure 20 is a flow illustrating a procedure according to which the viewer operates in use;

25 Figure 21 illustrates a procedure by which the image processing apparatus is operative to delete an object from a scene;

Figure 22 is a flow diagram illustrating a procedure by which the image processing apparatus is operative to add an object to a scene;

5

Figure 23 is a schematic diagram of an object in a light field demonstrating the potential for computational complexity in considering diffuse reflection;

10 Figure 24 is a schematic diagram showing a portion of the surface of the object illustrated in Figure 23, illustrating a gathering step of a method of reducing computational complexity;

15 Figure 25 is a diagram of the portion of the object surface illustrated in Figure 24, demonstrating a "shooting" step of the method of reducing computational complexity;

20 Figure 26 is a view of the object illustrated in Figure 23, demonstrating a further method of reducing the computational complexity of considering diffuse reflection;

25 Figure 27 is a schematic diagram of image processing apparatus for defining an environment in virtual reality, in accordance with the specific embodiment of the

invention;

Figure 28 is a side view of a head mounted display for use in the apparatus illustrated in Figure 27; and

5

Figure 29 is a view of the display apparatus of the head mounted display illustrated in Figure 28.

Figure 1 is a block diagram showing the general arrangement of an image processing apparatus according to an embodiment of the invention. In the apparatus, there is provided a computer (2), which comprises a central processing unit (CPU) (4) connected to a memory (6) operable to store a programme defining the sequence of operations of the CPU (4) and to store object and image data used in calculations by the CPU (4).

15

An input device (8) is coupled to an input port (not shown) of the CPU (4). The input device (8) may comprise, for example, a keyboard and/or a position sensitive input device such as a mouse, tracker ball, or a digitizer tablet and stylus etc.

20

A frame buffer (10) is also coupled to the CPU (4), the frame buffer (10) comprising a memory unit (not shown) arranged to store image data relating to at least one image, for example by providing one (or several) memory

25

location(s) per pixel of the image. The value(s) stored in the frame buffer (10) for each pixel defines the colour or intensity of that pixel in the image.

5 In the present embodiment of the invention, an image is represented as a two-dimensional array of pixels, which can conveniently be described in terms of Cartesian co-ordinates. The position of the given pixel can be described by a pair of x, y co-ordinates. The frame
10 buffer (10) has sufficient memory capacity to store at least one image. If the image has a resolution of 1000 by 1000 pixels, the frame buffer (10) should include 10^6 pixel locations, each location being addressable directly or indirectly in terms of pixel co-ordinates x,y.

15

A video display unit (VDU) (12) is coupled to the frame buffer (10). The VDU (12) is operative to display the image stored in the frame buffer (10) in a conventional manner. For instance, if the VDU (12) displays images
20 in a raster scanning manner, the x co-ordinate of a pixel maps to the distance along a line of the scanned display, and the y co-ordinate of the pixel maps to the number of the line.

25 Also coupled to the frame buffer (10) is a video tape recorder (VTR) (14) adapted to receive a video tape (15). Another image recording device, such as a paper printer

a 35mm film recorder or a recordable compact disc could be provided in addition or in the alternative.

A mass storage device (16), such as a hard-disk drive, is coupled to the memory (6). The mass storage device (16) has a high data storage capacity, suitable for storing data to which instant access is not required. Moreover, a disk drive (18), operable to accept removable data storage media such as a floppy disk (20) is coupled to the memory (6). The disk drive (18) is operable to transfer data stored on the floppy disk (20) to the memory (6).

Figure 2A illustrates a scene (22) in respect of which it would be desirable to model light propagation using the apparatus of the embodiment of the invention described herein. The scene (22) consists of a room bounded by left and right side walls (24, 26), back and front walls (28, 30), a floor (32) and a ceiling (34). The front wall (30) and the ceiling (34) are shown as transparent in Figure 2A for reasons of clarity. The walls, floor and ceiling together define a room which, in this embodiment, is substantially cubic in shape.

A substantially square window (36) is set in the left-hand side wall (24), with edges parallel to the edges of the wall (24). A rectangular mirror (38) is mounted on

the back wall (28) with the longer sides of the rectangular mirror (38) being substantially vertical.

5 A standard lamp (40) of conventional size stands on the floor (32), substantially in the corner defined by the left-hand side wall (24) and the back wall (28). A chair (42) is situated substantially in the centre of the floor (32). The chair (42) is of simple construction, having
10 in the room such that its back is substantially parallel with a line constructed between the back left corner of the room (in which the standard lamp (40) stands) and the front right corner of the room. The chair (42) faces the front left corner of the room.

15

In Figure 2A, an eye (44) is placed at a viewpoint at the front of the scene (22) and slightly to the left of centre. As is well-known from conventional optics, light enters the eye (44) and is refracted by various elements
20 of the eye (44), to allow a focused image to be formed on the retina.

In the scene (22), rays of light travel in an infinite number of directions. However, only those rays which
25 enter the pupil of the eye (44) are experienced by the observer. For example, a ray (46) is shown travelling from the window (36) (transmitting light from a

theoretically infinitely distant light source), directly to the eye (44). Another ray (48) is shown travelling directly from the standard lamp (40) to the eye (44).

5 However, a substantial amount of light will not reach the eye (44) directly from light sources, but will be reflected off objects within the scene (22). For example, light radiated from the standard lamp (40) will hit the left-hand wall (24), the floor (32) and the
10 right-hand wall (26), and be reflected towards the eye (44). Rays (50a, 50b, 50c) respectively illustrate these three sets of circumstances. It will be understood that light radiated from the lamp will also be reflected towards the eye from the back wall (28), the front wall
15 (32) and the ceiling (34), but rays corresponding to those paths are omitted from Figure 2A for reasons of clarity.

Moreover, rays (52a, 52b) in Figure 2A represent light
20 travelling from the window (36) to the eye (44) via a reflection on the back of the chair (42). Rays (54a, 54b) represent light travelling from the lamp (40) to the eye (44) via a reflection of the seat of the chair (42).

25 The walls, floor and ceiling (24-34) and the chair (42) have diffusely reflecting surfaces. Consequently, a ray of light incident on one of those objects will reflect

in all directions forming an acute angle with the surface normal at the point of incidence. Unit vectors representing those directions define a hemispherical volume of unit radius, with centre at the point of incidence, and bounded by a plane tangential to the surface of the object at the point of incidence.

As a result, an observer of reflected light from one of these objects will see a representation of the object in question, rather than a sharp reflected image of the ultimate source of the reflected light. Also, some of the light incident on each of these objects will be absorbed thereby. The rate of absorption is normally frequency dependent, and the absorption is associated with interpretation by the observer of colour of the object in question.

In contrast, the ray (56) illustrated in Figure 2A, which is firstly diffusely reflected from the back of the chair (42) towards the mirror (38), is then specularly reflected from the mirror towards the eye (44). Specular reflection is that which behaves in accordance with the ideal laws of reflection, as described in the introduction above. By observing other rays which similarly diffusely reflect from the various objects in the scene to the mirror and then to the eye, a reflected image of the scene is formed in the eye.

Referring now to Figure 2B, the same scene (22) is illustrated, with the eye of the observer in a new position. The eye is now referenced with the numeral (44'). The objects within the scene (22) described above are given the same reference numerals as before, in view of the fact that they are unchanged. However, the scene (22) is now being viewed from a different position, and so different rays of light will enter the eye (44'). For example, a ray (58) travels directly from the window (36) to the eye (44'), and a ray (60) travels directly from the lamp (40) to the eye (44'). Furthermore, rays (62a, 62b) travel from the lamp (40) via diffuse reflection of the chair (42) to enter the eye (44'). A ray (64) will travel from the lamp (40), diffusely reflect from the back wall (28), and enter the eye (44'). A ray (66) from the lamp (40) will specularly reflect from the mirror (38) and enter the eye (44'). That ray (66) is representative of rays from the lamp which reflect from the mirror (38), which will contribute to the construction of a reflected image of the lamp within the eye.

From the foregoing description of the two views of the same scene, it will be seen that in order to observe the same scene from two different views, one must obtain information concerning the radiance of light associated

with two different sets of rays.

In the real world, light exists independently of a viewer, and behaves in accordance with physical laws such as concerning reflection, refraction and diffraction. Where a view of a scene is observed in the real world, those rays which enter the aperture of a viewer are taken into consideration, and focused to produce an image. The preferred embodiment of the present invention aims to emulate that situation, by modelling a complete pattern of light behaviour within a scene, without reference to a particular viewpoint.

This objective is not achievable in practice, because any scene contains an infinite number of potential light propagation directions. Therefore, the present invention as illustrated by the preferred embodiment models a light field consisting of a finite number of potential carriers of light radiance, referred to hereinafter as "rays".

20

Thereafter, objects can be rendered into the light field by considering the intersections of the rays in the light field with the objects. Once all objects have been rendered into the light field, light can be propagated along the rays from objects designated as light emitters, under interruptions such as reflections, refractions and defractions can be computed at the intersections of the

rays with the objects.

The preferred embodiment is significantly different from the technique described by Levoy and Hanrahan and Gortler et al, both of which are directed to the construction of 3D images from interpolation between 2D images held in, for instance, a frame buffer. In contrast, the preferred embodiment described herein is concerned primarily with the behaviour of light within the light field, and considers the characteristics of the objects rendered into the light field only when calculating interactions at intersections.

Series of 2D images need not be stored with the present invention; an image at a particular viewpoint in the light field can be constructed in a straightforward manner.

Various different light fields could be defined, but a preferred light field consists of a plurality of parallel sub-fields. Each parallel sub-field consists of a light region, of generally cuboid shape, between two sides of which extend a plurality of parallel rays. The plurality of parallel rays is arranged in a grid to enable straightforward referencing thereof. The centre of each parallel sub-field is defined as its origin, and the parallel sub-fields are overlaid so as to have a common

origin. The parallel sub-fields are oriented in different directions in relation to each other, so that a light field is constructed which includes rays offset across a region, and oriented in a variety of directions in three-dimensional space.

With reference now to Figure 3A, a parallel sub-field (70) comprises a cubic volume in x, y, z Cartesian space defined by:

$$\begin{aligned} -1 &\leq x \leq 1 \\ -1 &\leq y \leq 1 \\ -1 &\leq z \leq 1 \end{aligned}$$

A rectangular grid is imposed on the x, y plane comprising n sub-divisions of equal width in each of the x and y directions. The width of each sub-division in each of the x and y directions is:

$$\frac{2}{n}$$

Discrete co-ordinates can be assigned to the centre of these sub-divisions, imposing a discrete co-ordinate grid on the x, y plane within the parallel sub-field 70. The grid can be referenced by co-ordinates (i, j) . The real co-ordinate on the x, y plane corresponding to (i, j) is therefore:

$$(x_i, y_i) = \left(\frac{2i+1}{n} - 1, \frac{2j+1}{n} - 1 \right); i=0, 1, \dots, n-1; j=0, 1, \dots, n-1$$

Each grid reference has associated therewith a ray (72) parallel with the z axis and spanning the cubic volume. Therefore n x n rays can be defined with reference to
 5 co-ordinates (i, j).

In that way, a parallel sub-field (70) is defined which contains a plurality of rays (72) in the vertical direction, which can be used to model light travelling
 10 in either direction along each ray in that set of rays.

With reference to Figure 3B, the cubic volume (70) previously described is illustrated rotated through an angle ϕ from the vertical about the y axis. In that way,
 15 each ray of the n x n array of rays (72) is oriented at an angle ϕ from the vertical and when resolved into the x, y plane is parallel with the x axis.

Figure 3C illustrates the cubic volume containing the array of rays further rotated through an angle θ about the z axis. In that way, each of the rays of the n x n array is oriented at an angle ϕ from the vertical and, when resolved into the x, y axis, is at an angle θ relative the x axis.

All possible directions of rays can be described using the following ranges:

$$\begin{aligned} 5 \quad & \phi = 0 ; \theta : \text{don't care} \\ & 0 < \phi < \pi/2 ; 0 \leq \theta < 2\pi \\ & \phi = \pi/2 ; 0 \leq \theta < \pi \end{aligned}$$

10 It should be noted that at $\phi = 0$, the direction of the rays (72) is independent of the value of θ , and that at $\phi = \pi/2$, θ need only pass through a half revolution in order to cover all possible directions.

15 In order to implement the arrangement on a computer, θ and ϕ must be discretised. m is a predetermined number of sub-divisions of 2π , representing the level of resolution which the apparatus is to use. Then, defining the discretised variable u, v corresponding to continuous variables θ, ϕ :

20

$$\theta = u \frac{2\pi}{m}, \quad u = 0, 1, \dots, m-1$$

$$\phi = v \frac{2\pi}{m}, \quad v = 0, 1, \dots, \frac{m}{4}$$

Therefore, by substituting these equations into the range definitions of ϕ and θ and making u, v the subject thereof, the ranges of values which u and v can take are

as follows:

$v = 0; \quad u : \text{don't care}$
 $v = 1, 2, \dots, (m/4)-1; \quad u = 0, 1, \dots, m-1$
 5 $v = m/4; \quad u = 0, 1, \dots, (m/2)-1$

Since the above formulae refer to the quantity $m/4$, m must be an integer multiple of 4 for the above arrangement to be consistent with u and v being integers.

10

Defining rays over the ranges set out above, a complete light field can be represented by the co-ordinate system (i, j, u, v) . This can be stored in the computer memory (6) as a 4D array.

15

A scene can be defined within the light field constructed of the arrays of rays. It is convenient to confine the scene for example to a unit cube space defined by the ranges:

20

$- 0.5 \leq x \leq 0.5$
 $- 0.5 \leq y \leq 0.5$
 $- 0.5 \leq z \leq 0.5$

25 That is because, any orientation of the cubic volume of the parallel sub-field (70) will fully enclose the scene. That means that no part of the scene is omitted from

coverage by rays in any particular direction. In fact, since the cubic volume (70) has smallest (edge) dimension being 2, any scene with longest dimension no greater than 2 can be accommodated within the cubic volume (70). For instance a scene consisting of a cube centred at the centre of the cubic volume can be at most

$$\frac{2\sqrt{3}}{3}$$

along each edge, since the volume diagonal of such a cube is 2 units in length. The volume diagonal of a cube is its longest dimension.

10

Assuming that the real ray (48) in Figure 2A matches with a ray in the light field. It can be identified by a start point (x_1, y_1, z_1) and a direction (dx, dy, dz) . (θ, ϕ) or (u, v) can then be found using trigonometry on direction (dx, dy, dz) . Matrix rotation of the ray (48) will rotate the ray so that it becomes vertical (parallel with the z axis). Then, the (i, j) co-ordinates can be found from the point of intersection of this transformed ray with the x, y plane and the previously stated definition of i, j in terms of x, y .

20

In the event that the real ray (48) does not match with a ray (72) of the light field, it is necessary to find a ray within the light field which best approximates the

real ray. It will be appreciated that for any accepted measurement of error and for any accepted error bound, m , n , the discretisation variables, can be selected. Firstly, once the real ray directions has been converted
 5 into direction coordinates (θ, ϕ) or (u, v) , the direction coordinates are "rounded" to the nearest provided value. In the case of (u, v) , this will be the nearest integer. Secondly, once the actual intersection of the real ray, rotated back to the vertical, with the
 10 x, y plane has been found, the co-ordinates of the intersection can be "rounded" to the nearest provided value of (x, y) or (i, j) . In the case of (i, j) , this will be the nearest integer.

15 In reverse, a given set of co-ordinates (i, j, u, v) defines a ray (72) which may be within the scene. u and v can be used to calculate (θ, ϕ) for that ray (72). Therefore, $(x_1, y_1, -1) R_y(\phi) R_z(\theta)$ and $(x_1, y_1, +1) R_y(\phi) R_z(\theta)$ give the end points of the ray in real co-ordinates,
 20 where R_y and R_z are standard three dimensional rotation matrices about the y and z axes respectively. This is useful for identifying position of a point lying on the ray within the scene.

25 The scene (22) of Figures 2A and 2B is illustrated again in Figure 4. A particular plane (74) within the scene is identified by dash lines; the position of the plane

(74) is defined by a horizontal line along the left-hand side wall (24) substantially one third of the height of the wall from the top of the wall, and a horizontal line along the right-hand side wall (26) substantially at the foot of the right-hand side wall. The plane (74) intersects the standard lamp (40) and the back of the chair (42). The plane (74) will contain a large number of rays, depending on the values of m and n selected. However, Figure 5 shows a selection of these rays (72) in three different directions. All other rays have been omitted for reasons of clarity. The portion of the scene intersected by the plane (74) is shown, with the window (36), the mirror (38), the standard lamp (40) and the back of the chair (42) illustrated. Moreover, a theoretical model of an eye, hereinafter referred to as a virtual eye (44) is shown, with a pupil (76), a lens (78), a focal plane (80), an image plane (82) and an optical axis (84).

One particular ray (86) is shown, which passes along the plane through the standard lamp (40) and the back of the chair (42). This is illustrated in more detail in Figure 6. The ray, when transformed back to the vertical, extends from $z = -1$ to $z = +1$, with the various objects intersected, namely the left-hand side wall (24), the standard lamp (40), the back of the chair (42) and the right-hand side wall (26) placed thereon at the

appropriate place. All points along the ray can be assigned a position by referring back to the z axis.

Figure 7 illustrates the segment of the ray between the side walls (24, 26) which is the part of the ray which is of most interest, being as it is part of the scene (22).

In order to render an object of the scene into the light field, firstly a ray passing through an object must be found. Then, when such a ray (86) is found, all of the intersections of that ray with the object must be identified by position. That position is expressed as a value t in the parametric equation:

$$p(t) = p + t (q-p) \quad 0 \leq t \leq 1$$

where p and q correspond with intersections of the ray (86) with the limits of the scene, i.e. side walls (24, 26). Therefore, as intersections are found, the ray (86) has an associated set of intersections $[0, t_1, t_2, \dots, t_k, 1]$ where the t_i are parametric values corresponding to the positions of the intersections with objects. The real co-ordinate position of each intersection can be found from $p(t_i)$.

In order to keep account of intersections of the ray, the intersections are arranged in an interval list. Interval lists for all rays of the entire light field are arranged in a four-dimensional array Ray Interval (i, j, u, v),
5 i.e. expressed in terms of the same co-ordinates as the rays themselves. Each interval list is initially set to null, before intersections are found and arranged therein. A null interval list represents the interval $[0, 1]$, i.e. the full length of the ray within the bounds
10 of the scene.

Each entry in the interval list includes not only the t value, but other data concerning the nature of the intersection as well. All of the data is collected in
15 a structure called a T-Intersection, having fields for the t -value, the object intersected, the direction of potential radiance from the intersection, a radiance value r (initially set to zero), and an unshot radiance value u (initially set to zero).

20

The direction is set to be either "right" or "left". The "right" direction corresponds with the direction along the ray from $z < 0$ to $z > 0$ when the ray is transformed back to the vertical and "left" is in the opposite direction.

25

The last two quantities may be vector quantities if the system is arranged to consider colour, polarisation or

other characteristics of light within the system which need to be described. The structure of the T-Intersections is illustrated in Figure 3. It may also be advantageous to include a field for the "normal vector" at the point of intersection with the object, though this is not essential since the normal can always be computed by reference to the point of intersection and the data defining the object.

10 The T-Intersections should be placed in a data structure which allows for ease of implementation and relatively constant look-up time.

In this embodiment, a standard recursive binary tree is used, known hereinafter as an interval tree. The interval tree corresponding to the ray (86) illustrated in Figure 7 is illustrated in Figure 9.

In the general case, each node of the interval tree corresponds to a T-Intersection and further comprises two child nodes, called left_Tree and right_Tree. left_Tree and right_Tree are also interval trees. When an object is rendered into the light field, and a first intersection is found with the ray under consideration, the intersection is loaded into the previously empty interval tree for that ray. The interval tree now comprises one node, and the two child nodes are null.

As a second T-intersection is loaded into the interval tree, the t-value contained in that second T-Intersection is compared with the t-value contained in the T-Intersection of the first node. If the t-value of the second T-Intersection is lower than the t-value of the first T-Intersection, then the second T-Intersection is placed in the left-Tree child node of the first node. Otherwise, the second T-Intersection is placed in the right Node child node of the first node. In that way, an interval tree is constructed which is sorted in respect of the t-values, which binary tree can be used with ease to search for T-Intersections by t-value.

A third intersection can be added in the same way - if a child node is full then comparison is made with the t-value of the contents of that child node. Progress is made down the branches of the tree until a null child node at an appropriate position is found.

An intersection where transmission of light is predicted from the characteristics of the object cannot be expressed purely in terms of having a direction of potential light radiance from the intersection. In that case, two T-Intersections must be created, namely $T_1 [t_1, \text{object, left, } r_1, u_1]$ and $T_2 [t_2, \text{object, right, } r_2, u_2]$, where $t_1 = t_2$. Clearly, sorting by t-value is

insufficient to find a valid position for T_1 and T_2 on a binary tree, so it is established that in the event that $t_1 = t_2$, the direction (i.e. left or right) is taken into account and T_1 will reside to the left of T_2 .

5

In the example illustrated in Figure 7, the ray (86) is parameterised such that $t = 0$ at the left-hand wall (24) and $t = 1$ at the right-hand wall (26). The T-Intersections along the ray (86) are identified by subscript, with T_1 to T_4 representing intersections with the chair (42) and T_5 and T_6 representing intersections with the lamp (40). T_7 and T_8 represent the intersections with the side walls (24, 26). Accordingly, the T-Intersections have the following attributes:

15

$T_1 = (0.50, 42, \text{left}, 0, 0)$

$T_2 = (0.52, 42, \text{right}, 0, 0)$

$T_3 = (0.70, 42, \text{left}, 0, 0)$

$T_4 = (0.72, 42, \text{right}, 0, 0)$

20

$T_5 = (0.07, 40, \text{left}, 0, 0)$

$T_6 = (0.15, 40, \text{right}, 0, 0)$

$T_7 = (0.00, 24, \text{right}, 0, 0)$

$T_8 = (1.00, 26, \text{left}, 0, 0)$

25 If those T-Intersections are found in that order, they will be loaded onto the binary tree illustrated in Figure

9.

Once T-Intersections are loaded into the binary tree, look-up time is dependent on the logarithm of the number of T-Intersections. This is an advantageous arrangement because look-up time in the interval tree will not increase significantly as an increasing number of intersections are loaded thereon. Moreover, as will become apparent from later description, at times it is necessary to find a T-Intersection adjacent a given T-Intersection in a given direction. It is relatively straightforward to find that T-Intersection using the binary tree in its conventional way.

Once all rays have been found which intersect objects, and all intersections have been loaded onto relevant binary trees, the rendering of the scene into the light field is considered complete.

Thereafter, radiance must be added to the light field, by taking into consideration any objects within the field which emit light. In the present example, the window (36) and the standard lamp (40) are considered to be light emitters.

25

Taking into consideration the standard lamp (40), all of those rays which intersect the standard lamp are found.

When a ray (86) is found, a T-Intersection T_5 is identified on the ray as intersecting with the standard lamp (40). Radiance in accordance with the light emission characteristic of the standard lamp (40) is added to the data structure, and unshot radiance equal to the change in radiance is also added thereto. T-Intersection T_6 is then found and treated in the same way, and thereafter, all rays intersecting the standard lamp (40) are treated in the same way.

10

Moreover, although not illustrated in Figure 5, rays intersecting the window (36) are treated in the same way with respect to light emitted thereby, or more properly transmitted therethrough. In the interests of simplicity, it is more straightforward to consider the window as an object with light emitting properly than to render the sun into the scene.

Obviously, if it is desired to represent further objects on the other side of the window, those objects must be rendered into the light field as well. This may require some scaling of the scene in order to ensure that the whole scene is enclosed.

Once all light emission has been added to the light field, each object is considered to establish whether it is intersected by a ray which carries unshot radiance.

In the example shown in Figures 5 and 7, the back of the chair (42) is in receipt of unshot radiance along segment (T_6 , T_1). At that point, the unshot radiance is considered to be emitted light from that point on the chair, and it is transmitted through diffuse reflection along all rays emanating from on or near T-Intersection T_1 on ray (86).

In the same way as a real ray is unlikely to match exactly a ray in the light field, it is unlikely that exact coincidence with other rays will occur at intersection T_1 , and so approximations will be necessary. In fact, in each desired direction, the closest ray is selected, and the diffusely reflected radiance and unshot radiance is added to each such ray. Those diffuse reflections are identified by arrows (88, 90, 92, 94) in Figure 5. Moreover, some radiance will be reflected back onto ray (86) towards the standard lamp (40). Rays (88, 90) are shown despite the fact that they do not coincide with illustrated sets of rays; they will each coincide with another ray not illustrated but within the set of rays in the light field which are contained in the illustrated plane (74). Other diffusely reflected rays will also be identified in other directions not contained in the plane (74).

Once all unshot radiance incident on the chair (42) is dealt with in this way, the next object to be in receipt of unshot radiance is considered. For example, the mirror (38) is now in receipt of unshot radiance as a result of reflected light from the chair along the ray (92). However, in that case the mirror is a specular reflector, and so only one true reflected ray exists. That true reflected ray is illustrated as a broken line denoted by reference numeral (96) in Figure 5. In the example embodiment, no ray within the light field exactly coincides with true ray (96). Therefore, the parallel sub-field (70) in a direction nearest to the direction of the true ray (96) is identified and the ray (98) in that parallel sub-field (70) closest to the true ray (96) is identified. Then the unshot radiance along the ray (92) is reflected along the best fit reflected ray (98). Therefore, by iterating through all objects in receipt of unshot radiance, radiance can be added at T-Intersections in accordance with the distribution of light in the scene. In fact, the objects marked as being in receipt of unshot radiance are considered in decreasing order of the amount of unshot radiance received by each object. In that way, objects having most effect on illumination of the scene are dealt with first. The above technique is carried out without regard to any selected viewing position at this point.

As noted previously, a virtual eye (44) is illustrated in Figure 5. In order to view the scene from this point, all rays identified as entering the pupil (76) are considered to have entered the virtual eye (44). The direction of each ray and its position relative the optical axis (84) are identified, and a lens equation defining the structure and position of the lens (78) is applied thereto. A lens equation is a vector equation which identifies the trajectory of a refracted ray from a given trajectory of an incident ray.

Thereafter, the radiance associated with that ray at that point along the ray is recorded in an array associated with pixels or sub-pixels in order to build up an image. Smoothing functions can be applied to the image so that any parts thereof which are not fully constructed having regard to the number of rays entering the pupil (76) can be filled. It will be appreciated that the step of building up an image is carried out independently of the calculation of the characteristics of the light field with object and light emitters rendered therein, and so that use of the light field is entirely view-independent.

The apparatus of the particular embodiment of the present invention will now be described in more detail with reference to Figures 10, 11 and 12.

Figure 10 is a basic block diagram illustrating operational modules of the computer (2) illustrated in Figure 1.

5 A user interface (100) comprises a facility for the interaction of an user with the apparatus. By virtue of the user interface (100), a user can identify the objects which are to be placed in a scene, and their characteristics such as light emission, absorption,
10 colour and/or position etc. Coupled with the user interface (100) is light field (LF) computation unit (102). The LF computation unit (102) is operative to define a light field in terms of the four-dimensional co-ordinate system defined above, to render objects into a
15 scene within the light field and to activate light emission such that radiance is distributed through the light field.

A viewer (104) is coupled with the LF computation unit
20 (102) so as to obtain data therefrom concerning radiance values along rays (72) of the light field. The viewer is operative to convert those radiance values into a focused image, and to process that focused image into a form which can be viewed. The viewer (104) is linked
25 with the VDU (12) as described with reference to Figure 1.

Figure 11 comprises a block diagram showing the components of the LF computation (102). The LF computation comprises a preordained LF which cannot be modified by the action of the user. Via the user interface (100), described with reference to Figure 10, a user defines, or calls up, a predetermined, scene definition file (108). That scene definition file contains information concerning the nature of objects to be placed in the light field (106). Those object definitions are placed in an objects file (110).

The LF computation unit (102) further comprises an intersection data computation unit (112) which considers each object in turn from the object file (110) in respect of the preordained LF (106), to establish the position of intersections along each intersected ray of the LF. The intersection data computation unit (112) is operative to produce an intersection data table (114) for all rays within the preordained LF (106).

20

The LF computation (102) also comprises a light computation unit (116) which makes reference to the objects file (110) to identify light emitting objects, and refers to the preordained LF (106) and the intersection data table (114) to render light through the LF. Light is rendered through the LF, as described previously, by up-dating data within the intersection

25

data table (114).

Figure 12 illustrates in more detail the components of the viewer (104). The viewer comprises a file of
5 predetermined viewers (118) (such as a human eye, a regular camera lens, fish eye lens, wide-angle view, zoom etc) which can be selected and/or modified by interface with the user through the user interface (100). The user interface (100) also provides a facility for specifying
10 the position and orientation of the viewer (104) within the LF. The viewer (104) further comprises a viewed image computation unit (120) which makes reference to the selected viewer lens and the intersection data table (114) of the LF computation (102). The viewed image
15 computation unit (120) is operative to produce an image in accordance with a lens equation describing the characteristic of the selected lens. It will be appreciated that a selected viewer (118) could include a sequence of lenses which, in combination, provide a
20 desired optical effect.

Data corresponding to that image is passed to a data conversion component (122), such as for converting the data into raster scan data for use with a VDU. Up to
25 that point, the level of resolution of the image can be defined to suit the resolution of the light field, and need only be converted into a fully pixellated image at

the final stage. The converted image is then output to the VDU (12).

A plurality of viewers (104) can be defined, either to deliver different images to different VDU's or, in the case of two viewers being provided, to deliver stereoscopic images to screens in a headset for use in Virtual Reality applications.

Figures 13 to 19 describe procedures performed by the LF computation (102) during operation thereof. Figure 13 describes the main procedure of the LF computation (102).

On commencement of the procedure, the LF is set up by the SET UP THE LF procedure described later. Once the LF is set up, objects are rendered into the LF by the RENDER OBJECTS INTO LF procedure (204) to be described later.

Once all objects have been rendered into the LF, those objects which are light emitters are activated, by means of the ACTIVATE LIGHT EMITTERS IN LF procedure (206) to be described later. Then, once the light emitters have been activated, radiance is emanated through the LF by means of the COMPUTE REFLECTIONS IN LF procedure (208) described below. Once that procedure has been completed, the LF is fully set up with objects rendered therein and light emission activated.

The SET UP LF procedure (202) will now be described with reference to Figure 14. Firstly, a light region (70) is defined as a grid of n by n parallel rays. The rays are arranged in x, y, z space parallel with the z axis and the light region is bounded by the following constraints:

$$\begin{aligned} & -1 \leq x \leq 1 \\ & -1 \leq y \leq 1 \\ & -1 \leq z \leq 1 \end{aligned}$$

Following that step, the light region is cycled through a predetermined number of orientations of ϕ and θ , wherein ϕ is the angle of the rays from the z axis and θ is the angle which the rays cover when resolved into the x, y plane defined with the x axis. In order to cover all possible directions of the rays, ϕ and θ must pass through the following range:

$$\begin{aligned} \phi &= 0 \\ 0 < \phi < \pi/2 ; 0 \leq \theta < 2\pi \\ \phi &= \pi/2 ; 0 \leq \theta < \pi \end{aligned}$$

The step "generate set of equispaced rays parallel with said normals" comprises defining the rays as look-ups into a four-dimensional array of interval trees as described above with reference to Figure 9.

The RENDER OBJECTS INTO LF procedure (204) identifies whether any objects are to be rendered into the LF. This is carried out by calling the objects file (110). If there are no objects to be rendered into the LF, the
5 procedure returns to the main procedure. Otherwise, the objects file (110) is called to obtain the parameters defining the object. Those parameters consist of the shape definition, which may be in terms of its geometry, or a skeleton, and the surface characteristics of the
10 object. Those surface characteristics may consist of the light absorption characteristics, and the bi-directional reflectance distribution function (BRDF). The object parameters may also include details of any light emission characteristics of the object. At this point, however,
15 only the geometry of the object is taken into account.

Following the obtaining of the parameters defining the object, the procedure calls a sub-procedure (210) namely, OBTAIN DETAILS OF INTERSECTS OF RAYS WITH OBJECTS (210).
20 When that sub-procedure (210) has completed, the procedure (204) repeats the inquiry as to whether any objects remain to be rendered into the LF. If no more objects require rendering into the LF, then the procedure returns, otherwise the procedure selects the next object
25 and repeats as necessary.

The OBTAIN DETAILS OF INTERSECTS OF RAYS WITH OBJECT

procedure (210) is now described with reference to Figure 19. Firstly, the procedure inquires as to whether any rays intersecting the object remain to be considered. If no more rays remain to be considered, the procedure
5 (210) returns to the previously described procedure (204). Otherwise an intersected ray is considered, and intersections of that ray with the object are successively loaded onto a binary tree for that ray, the intersection data consisting of the position of the
10 intersection on the ray, the identity of the object intersected, the potential direction of radiance from the intersection, and radiance and unshot radiance values which are initially set to zero. The procedure then loops around to establish whether any other rays remain
15 to be considered. If no more rays remain to be considered, then the procedure returns to the earlier described RENDER OBJECTS INTO LF procedure (204).

Once all objects have been rendered into the LF, in
20 accordance with procedure (204), the ACTIVATE LIGHT IMAGES IN LF procedure (206) is called. This procedure is described in more detail with reference to Figure 16. The procedure commences by considering an object with light emission properties. Thereafter, a sub-procedure
25 FIND A RAY INTERSECTING OBJECT AND INTERSECTION THEREWITH (212) is called. This procedure will be described later on.

Once a ray intersecting the object and its intersection therewith has been identified, the procedure (206) then defines an interval bounded by that intersection and the
5 next intersection along the ray from that object in the direction of predicted radiance defined in the data associated with the intersection. Then, the radiance in that interval is set in the data structure associated with the intersection under consideration. That radiance
10 is set in accordance with the light emitting properties of the object, which are called from the parameters defining the object as held in the objects file (110).

If the radiance in that interval, as defined in the
15 intersection data, is greater than a predetermined threshold, then the procedure (206) considers that it would be worthwhile computing any reflections from that radiance. As such, the procedure (206) then sets the unshot radiance in the intersection data in question to
20 correspond with the change in the radiance. The object at the other end of that interval is identified and that object is marked as being in receipt of unshot radiance. If the radiance in the interval is less than the threshold, then the unshot radiance is not set, and there
25 is no requirement to identify and mark the object at the other end of the interval. Thereafter, any other intersections of that ray with the object in question are

considered and radiance and unshot radiance are updated as required. Any other rays intersecting the object are then considered, and then any other objects with light emission properties are considered in the same way. Once
5 all objects having light emission properties have been processed in the same way, the procedure (206) returns.

The ACTIVATE LIGHT EMITTERS IN LF procedure (206) called the FIND RAY INTERSECTING OBJECT AND INTERSECTION
10 THEREWITH procedure (212). That procedure will now be described with respect to Figure 18. The procedure (212) firstly inquires as to whether the object is defined in terms of plane polygons. That information is contained in the objects file (118). If it is so described, a more
15 simplified procedure is followed. Each plane polygon is considered in turn, and rays passing through the vertices of the polygon are found. For each found ray, the intersection of that ray is found by interpolation between the vertices of the polygon under consideration.
20 Once that intersection is found, the procedure (212) returns.

The procedure of finding and dealing with rays passing through a plane polygon can be dealt with using a 2-D
25 fill algorithm as set out in Foley pp 92-99. Clearly, the use of values of z , to define positions of intersection of rays with objects, also allows use of the

invention alongside z-buffer techniques as also discussed in that publication, pp 68-672, to accommodate less photo-realistic but dynamic objects within an illustrated 2D image of the scene.

5

In an alternative case, an object may be described not in terms of plane polygons, but in terms of geometric shapes such as a sphere, an ellipsoid or composite thereof. In that case, a bounding box is defined around the object, the bounding box being axis aligned and of smallest size possible. A side of the box is considered, and a ray passing between the vertices of that side of the box is found. As each ray is found, a check is made as to whether the ray passes through the object. If it does not pass through the object, then a new ray is found and the same checks are carried out as before. Once a ray has been found, which passes through the object, then the intersections of the ray with the object are calculated with respect to the object shape definition. Once those intersections has been identified, then the procedure returns.

Finally, once all light emitters in the LF have been activated, and radiance has been added to all of the rays intersecting those light emitters, reflections through the LF are computed. That is carried out by calling the COMPUTE REFLECTIONS IN LF procedure (208) described now

25

in more detail with reference to Figure 17. Firstly, the procedure inquires as to whether there are any objects marked as having received unshot radiance. Objects will have been marked as a result of ray intersections having had radiance added to them above a predetermined threshold, and therefore corresponding unshot radiance will have been added thereto as well. An object which has been so marked is put under consideration, and is then unmarked. A ray intersecting the object is identified, along with its intersection. The procedure then searches through the data structure, which is conveniently in binary tree form, to identify the adjacent intersection along the ray in the direction of probable radiance as defined in the intersection data on another object. The adjacent intersection is checked to establish whether it contains unshot radiance. If it does not contain unshot radiance, then another ray intersecting the object in question is considered as before. Once a ray has been identified which has an intersection with the object and which is an adjacent intersection containing unshot radiance, a reflected ray is identified.

The step of finding a reflected ray requires references to the BRDF of the object, which is contained in the objects file (110). The BRDF may specify diffuse reflectance, specular reflectance or a combination of

both. In the case of diffuse reflectance, a plurality of reflected rays exist to be found, over the point of intersection of the incident ray under consideration with the object. With a specular reflection, the reflected ray may be a ray of best fit having regard to a true reflected ray as calculated with respect to the well-known laws of reflection.

Furthermore, the BRDF may specify that a cone of reflected rays be produced to correspond with a given incident ray. This effect represents imperfect specular reflection as described by Phong in CACM 18(6), June 1975, 311-317.

In respect of the or each reflected ray, the intersections of that ray with the object in question are identified. Then, the radiance in accordance with the BRDF is set in the intersection with the object in question, and that change in radiance is considered as to whether it is greater than the aforesaid predetermined threshold. If it is greater than the predetermined threshold, then the aforesaid adjacent object is marked as having received unshot radiance, and the aforesaid intersection with the object under consideration is updated so as to reflect the increase in unshot radiance. Otherwise, no consideration is taken of unshot radiance.

Once that has been carried out in respect of all reflected rays to be considered, the object is further considered to establish whether any other rays intersecting that object have received unshot radiance.

5 If no more rays are to be considered, then the light field is further considered to establish whether any objects remain which are marked as having received unshot radiance. Once all objects are unmarked, the procedure returns.

10

Once all objects have been considered for unshot radiance, the light field is fully defined with respect to the scene rendered therein. At that point, a complete data structure exists independent of any view position or the characteristics of any viewer. It would be possible to use the light field without reference to any viewer, for instance in the design of an art exhibition where even distribution of light throughout a region of the light field needs to be considered. The distribution of light in a gallery could be designed with the above-described arrangement, and the light incident on any particular wall could be monitored. In that way, bright patches or dark patches could be eliminated from a particular wall.

25

However the present invention is also particularly suitable for applications where a particular viewpoint

is required, such as is illustrated in Figure 5 with the simulated eye (44). A view can be constructed from the LF through the procedure now to be described with reference to Figure 20. This procedure stands alone from the procedure operating the LF computation, and can be controlled by the user interface (100). The viewer procedure commences by calling for a specification of an aperture of the viewer. This can be identified from the viewers file (118) under the control of the user (100). The specification of the aperture includes the actual diameter of the aperture plus its position in space and orientation. Once the aperture has been specified, a set of rays entering the aperture can be identified. This can be carried out using a procedure such as procedure (212) previously described.

As each ray is found which enters the aperture, the end point of that aperture on the image plane can be found. The ray can be considered as a vector which can be passed through a vector equation relating to the composition of the lens. The composition of the lens can be described with reference to the viewers file (118). The result of the vector equation is a point on an image plane (82). The image plane is defined as an array of elements, and the radiance associated with the segment of that ray, as called back from the previous intersection of the ray with an object, is mapped onto an element within that

array. It is possible that several rays will be incident on any particular element, and as each ray is incident on an element, the radiances are combined together.

5 Eventually, as all rays are considered, an intensity map will be formed on the image plane composed of elements, which can be processed by filtering and transformation from the array of pixels to form a pixellated image. In particular, if the lens structure is that of a relatively
10 simple box camera, the image will need inversion before it can be displayed correctly on a VDU.

The viewing procedure set up above can be relatively fast, and is independent of the level of complexity of
15 the image. In fact, it can be considered to be generated in substantially constant time from frame to frame as the viewer moves. With sufficient and reasonable processing power, that constant time could be reduced to such an extent that the viewer could be implemented as a real-
20 time viewer.

The above identified arrangement can also be used in a headset for virtual reality applications, with stereoscopic images being projected to the eyes. Both
25 images can be computed rapidly. Moreover, the angle of gaze of the eyes of the wearer of the headset. This is an important development which has previously required

significant computational power for its achievement.

Although the above apparatus has the potential for a view of a static scene to be changed in real-time, dynamic
5 changes to the scene itself in real-time may be rather more difficult to achieve. However, the apparatus can be combined with other apparatus to define an image of a static, background scene in accordance with the present disclosure, while the other apparatus produces an image
10 which can be changed dynamically, which image is superimposed over the image of the static scene. As noted earlier, the apparatus can readily be used to retrieve z-values for combination with a z-buffer method for rendering objects into a scene.

15

Moreover, the embodiment described above assumes that the light transmitting medium is non-participatory. In practice, media are rarely non-participatory, and so attenuating properties can be represented by means of
20 placing intersections at random along the rays in the light field. The concentration of those random intersections depends on the level to which the medium in question attenuates light transmitted through it. In that way, the scene can be illuminated taking account of
25 the attenuating properties of the medium. In fact, attenuation is commonly the result of dust particles in suspension in air, and so the above modelling technique

is a reasonable approximation of reality.

The preferred embodiment of the invention as described above has been described in relation to the modelling of a scene to which no changes are to be made following modelling. However, circumstances could arise in which it would be desirable to be able to delete an object from the scene, or to add an object to the scene. Figures 21 and 22 describe procedures by which these two actions can be achieved.

Firstly, Figure 21 illustrates a flowchart defining a procedure for deleting an object from a scene defined in the light field to which light has already been applied. As a first step, a ray is found which intersects the object to be deleted. Thereafter, an interval is identified along that ray, having its second T-Intersection being with the object in question and having direction of potential radiance being to the left. The second intersection is that intersection of the interval with higher t-value.

Then, along that interval, the radiance R is noted. The surface normal on the object to be deleted is found at that second intersection, and the redistribution of radiance along reflected rays is established. The redistribution is applied to the reflected rays by

subtraction therefrom, and the subtracted radiance is also noted as negative unshot radiance therein. Objects are marked where appropriate as being in receipt of that negative unshot radiance.

5

Then, the interval corresponding to the inside of the object to be deleted is found. The radiance along that interval is set to equal R. Finally, the interval having first intersection being with the object and having
10 direction of potential radiance being designated as to the right is found. The unshot radiance for that interval is set to R, and the object (if any) at the other end of that interval is marked as being in receipt of unshot energy.

15

Then, an enquiry is made as to whether any further rays are to be considered. If so, the above procedure is repeated in respect of further rays; otherwise, the intersections corresponding to deleted objects are
20 deleted from the interval trees relating to those rays. Then the previously described routine COMPUTE REFLECTIONS IN LF is called in relation to the objects marked as being in receipt of unshot energy.

25 Figure 22 is a flow diagram showing steps of the procedure designed to allow an object to be added to a scene rendered into a light field, after light emitters

have been activated.

Firstly, a ray intersecting the object to be added is found. Then, an interval along the ray with second
5 intersection being with the object and having direction of potential radiance being to the left is identified. The radiance R along that interval is noted, and the surface normal at the second intersection with the object is found. Rays along which that radiance R is to be
10 propagated are determined, and radiance and unshot radiance are added to the intersections of those rays with the object as necessary. Any objects in receipt of unshot energy are marked.

15 The interval along the ray corresponding to the inside of the object is identified, and the radiance therealong is set to zero.

Finally, the interval with first intersection being with
20 the object and having directions of potential radiance being to the right is found. The unshot radiance along that interval is set to be $-R$, and the object (if any) at the other end of that interval is marked to be in receipt of unshot energy. The fact that the unshot
25 energy received by the object is "negative" is not relevant; it is only important that there is a change in unshot energy which needs to be processed through the

light field.

Then, an enquiry is made as to whether any further rays are to be considered in relation to the object to be added, and if so, the above steps are repeated. If no rays remain to be considered, the procedure COMPUTE REFLECTIONS IN LF is called in respect of the objects marked as being in receipt of unshot energy, following which the ADD AN OBJECT procedure is completed.

10

The above two procedures are optional features, which can preferably be called from the user interface 100.

These two procedures can be combined in order to provide a procedure for moving an object. In the combined procedure, the union of the object in its original position and the object in its new position is put under consideration, and the rays intersecting this union are found. This is computationally less expensive than considering the deletion and the addition of the object as separate steps to be applied in turn.

The invention, as described by way of the above exemplary apparatus and procedure, is particularly advantageous in that it is capable of providing a globally illuminated scene which can be amended by adding, deleting or moving an object in the scene without the need to recalculate

the illumination of the entire scene.

Further and alternative means of listing the T-Intersections could be provided. For example, an array
5 could be defined for each ray and the T-Intersections could be loaded into this array by t-value. This has the disadvantage of needing to choose a maximum number of intersections along a ray in advance.

10 Alternatively, the T-Intersections could be stored in a standard linked list, or a doubly-linked list so that it is particularly easy to insert new elements or to delete elements from the list. The disadvantage of this is that there would be linear search time for any particular
15 element, compared to the logarithmic search time required for the binary tree method.

As a final alternative, it would be possible to divide the ray into N equal segments. If N is large (say 1000)
20 then any t value is approximated by an index into a particular segment of the ray. Hence, a one dimensional array of T-Intersections, indexed by the index number of the segment corresponding to the intersection, could be used. This has the advantage of constant look-up time
25 (so it is faster than the binary tree method for large number of intersections) but the disadvantage of approximation and memory requirements (since most of the

entries in the array would be null). For example, suppose that $N = 1000$, but that there are only twenty T-Intersections along a given ray. In that case, 980 of the possible entries in the array for that ray would be empty.

Two ways exist of overcoming the memory disadvantages of the last method. The first way is to use a run length encoding. In that case, the storage would be in the form of the number of null entries, followed by the actual non-null entries, and so on throughout the array. However, the look-up time for that array would be non-constant depending on the number and location of non-null entries of the array.

15

The second way of improving the final method is by packing many possible representations into words, rather than considering them as single array entries. For example, if $N = 1024$ (2^{32}), 1024 entries of "0" and "1" can be represented by 32 unsigned integers. Hence, instead of having an array of 1024 potential T-Intersections, an array of 32 unsigned integers each initialised to all zeros is provided. This is less memory intensive. Associated with each such integer is an ordered list of those particular T-Intersections that correspond to "1" entries in the word. So, given a particular t-value, the closest segment in the

25

representation can be found. The quotient of the index of the segment divided by 32 will give a particular array element in which the t-value is located. The remainder of the index of the segment divided by 32 will identify
5 the particular bit that must be set to 1 to represent this t-value. Finally, the T-Intersection can be stored in a linked list associated with this particular integer.

It will be appreciated that the technique described above
10 is computationally relatively more expensive when considering diffuse reflectors than when considering specular reflectors. That is because diffuse reflection produces a very large number of reflected rays from a single incident ray. In fact, if a diffuse reflector is
15 included in a scene and dealt with according to the method described above, the level of computation can become prohibitive. Therefore, the following procedure has been devised with a view to reducing the level of computation required for considering diffuse reflectors
20 within a scene.

For example, Figure 23 shows a perfectly diffusely reflecting object (230), a surface of which is intersected by first and second rays (232, 234) of a
25 light field as described above. The two rays (232, 234) also intersect another body (not shown) which is a light emitter. The two rays (232, 234) intersect a surface of

the diffusely reflecting object (230) substantially at the same point.

Therefore, when considering the receipt of unshot energy
5 by the object, and consequent reflection of light, according to the previously described method the first ray (232) is considered and its intersection with the diffuse reflector (230) is identified. Once the intersection has been found, reflected rays are
10 identified through the hemisphere centred at the intersection and defined by the region bounded by the plane tangential to the surface of the object at the intersection. Radiance and unshot radiance are applied to those reflected rays. Thereafter, the second ray
15 (234) is considered in the same way. However, since the intersection of the second ray (234) with the diffuse reflector is substantially the same as the intersection of the first ray (232) with the diffuse reflector, many, if not all, of the reflected rays in respect of the
20 second incident ray will be the same as those for the first incident ray. Accordingly, by considering reflection of the two rays separately, duplication of effort has taken place.

25 The following method has been devised with a view to avoiding such duplication of effort.

Figure 24 shows a schematic diagram of a portion of the surface of the diffuse reflector (230). The portion of the surface has been divided into a grid comprising a plurality of components (240). The two incident rays (232, 234) are illustrated as intersecting the object surface within one particular segment (240). Once the first ray intersection has been found, instead of reflecting the ray immediately, the energy associated with that incident ray is accumulated with respect to that element (240). Then, when the second incident ray (234) is calculated as intersecting within the same element, the energy associated with that ray is also accumulated in respect of that element (240). This so-called "gathering" step results in elements of the surface having accumulated energy associated therewith. The accumulation of energy is carried out according to a weighted sum of terms. Each term represents energy received from a different ray. The weighting of each term relates to the angle of incidence of the incident ray on the object surface, and the properties of the surface as defined by the BRDF for the surface.

Figure 25 illustrates the same portion of the object surface, in which a "shooting" step is identified. In this step, the accumulated energy for the surface element (240) is reflected to all identified reflected rays from the surface element (240). In that way, the

computationally expensive reflected energy calculation is carried out only once per surface element (240). Therefore, the level of computation required for calculating diffuse reflection can be reduced.

5

In fact, the above example of the proposed method of reducing computation is not practically advantageous, since it relies on maintaining a data structure relating to surface elements. According to the specific
10 embodiment of the invention as described above, no records are kept within the definition of the light field of surface geometry of the objects of the scene, and so there is no obvious location for storage of such data.

15 Instead, it is proposed that accumulated energy for an element be stored in an additional field within the structure of the T-intersection associated with that element and relating to a ray of the light field substantially coincident with the surface normal of the
20 object surface at that element of the surface. Practically, it is not possible to add the field only to the set of T-intersections corresponding to surface normals, and so the field is included as part of the structure of T-intersections in general. For all T-
25 intersections apart from those corresponding to surface normals, the accumulated energy field is initialised to zero. Therefore, as shown in figure 26, the T-

intersection associated with the ray coincident with the surface normal N at the point of intersection of the incident rays (232, 234) has stored therein the accumulated energy related to the radiance of the
5 incident ray.

Thereafter, the accumulated energy is released onto the reflected rays calculated in accordance with the BRDF for the object at that point.

10

The present invention has several applications, some of which have been identified throughout the description. Other applications will now be described.

15 Firstly, a model of a building can be rendered into the light field, and illuminated in a desired manner, to verify the quality of an architectural design. Moreover, lighting can be designed within that rendered building, for use in production of artistic works, such as
20 exhibitions, stage plays, or film sets.

Secondly, the ability to interchange viewers enables the use of the invention in the design of a sequence of shots for a television or film production. The invention would
25 be suitable to simulate in advance of constructing a film set the appearance of that set under various layout and lighting conditions, and with different types of camera

lens.

Thirdly, the light field is a digitally encoded representation of a scene, through representation of the light within the scene. A digital encoding is ideal for compression and subsequent transmission, for instance, for broadcast television. A local set top box, hardwired to decode the transmitted digital encoding and capable of extracting images from the light field defined thereby could be used to display games, or other entertainment to people in their homes. The digital encoding could also be transmitted across the internet.

Fourthly, in the case that two image are provided of views of the light field, those two images can be transmitted to a head mounted display. In that way, stereoscopic images can be formed, which can provide a highly immersive virtual reality environment.

In Figure 27, a block diagram is shown of an image processing apparatus having all of the components of the image processing apparatus illustrated in Figure 1.

However, in this case, the central processing unit (4) is operative to produce two images of the light field embodied therein. Those two images are produced by two viewers, having parallel principal optical axes a

suitable interpupillary distance apart, which interpupillary distance can be adjusted by a user.

The images produced of those viewers are delivered to the frame buffer (10) from the central processing unit (4).
5 Coupled with the frame buffer (10) is a CD manufacturing suite (250) operative to produce compact discs (252) on which are stored encoded data relating to images delivered from the frame buffer (10). In that way,
10 sequences of images from a light field in accordance with the invention can be stored on compact discs for viewing later on suitable viewing apparatus.

A head mounted display (HMD) (254) is provided, and is
15 operative to receive both sequences of images from the frame buffer (10). A second VDU 12' is provided, and the two sequences of images can be displayed on those two VDU's 12, 12'.

20 In this case, the input means (8) can include a haptic input device, such as is used commonly in virtual reality applications. In that way, progress of the user through the virtual reality environment can be controlled by the user through manipulation of the haptic input device, and
25 the two images produced by the central processing unit (4) can be adjusted accordingly. Moreover, the interpupillary distance can be measured within the HMD

(254), and that measurement can be supplied to the input (8). That interpupillary distance can be translated into the distance between the optical axis of the two viewers within the light field defined in the central processing unit (4) which produce the two images for display in the HMD (254).

The CPU (4) is further coupled to a data compressor (260) which is adapted to receive data relating to the definition of the light field in the CPU (4) and compressing according to a suitable compression algorithm. The data compressor (260) is coupled to a transmitter (262) operable to receive compressed data from the data compressor (260), and to transmit that data by modulation on electromagnetic carrier radiation. The transmitted data is received by a receiver (264) which is in practice implemented as a set top box for a television (266). The receiver (264) comprises a decoder (268), capable of receiving transmitted data and returning it to its uncompressed state, to retrieve the data defining the light field. The data defining the light field is then transferred to a viewer (270) of the receiver (264) which is of a similar construction to the viewers (104) illustrated in Figures 10 and 11 of the drawings. In that way, a definition of a 3D environment can be transmitted to a receiver for manipulation by a user.

With reference to Figure 28, the HMD (254) consists of a visor suitable to be placed over a user's head (272). A cable (274) connects the HMD (254) to the image processing apparatus previously described.

5

The HMD (254) conventionally comprises two display units (276) (as illustrated in Figure 29) on which the previously described stereoscopic images can be projected.

10

The present invention is particularly suitable for viewing an environment in stereo, since there is little computational effort involved in creating an image from a light field which has already been rendered.

15

Therefore, two views of the same light field can be created without excessive computation. The focal length of the viewers can be altered in response to monitoring the angle of gaze of the eye of the user, using monitoring means incorporated into the head mounted display (254), to establish objects within the scene which are of interest to the user. In that way, those objects can be brought into focus.

20

The description further comprises two annexes providing further explanation of the above described preferred embodiment of the invention.

25

ANNEX A

**The Virtual Light Field:
A New Computer Graphics Paradigm
for Global Illumination**

II Description of the Proposed Research

1. Abstract

This proposal is to support continued research into a new paradigm for 3D computer graphics. The research offers a solution to a central problem of computer graphics, which is the achievement of realistic global illumination at real-time frame-rates. It solves the problem of light transport through scenes that have many different types of emitting, reflecting and transmitting surfaces. It produces a rendering solution such that any scene, with no matter how many objects, can be rendered in (small) constant time on a typical workstation or PC. The new method does not require the standard graphics pipeline, although elements of this can be employed in an efficient implementation. The method does not require scenes to be represented in polygonal form. As well as offering an advance for the specific purpose of producing photo-realistic images in the context of a globally illuminated scene, the method offers a general paradigm for 3D computer graphics, one which does not require users to make a choice between 'real-time' or 'photo-realism'. Many of the research preoccupations in computer graphics today, the drive for real-time performance with very complex models, issues such as visibility culling or level of detail, are non-issues in this new paradigm. As with a new paradigm in any discipline it leaves aside many problems of the traditional way of thinking, but obviously has problems of its own. The method discussed in this proposal should be seen primarily as a different way to think about computer graphics. Although this research will obviously result in standard EPSRC research grant applications, it also is a 'mission' – to argue for this new way of thinking, and to build up a new research group based around its development and multifarious applications. This cannot be done in the context of a full administrative and teaching load. This proposal offers the opportunity for the UK to play a leading role in computer graphics research and development, which is currently dominated by US universities and companies. The proposal also fits with EPSRC goals, as expressed in recent calls from 'Human Factors in Virtual Reality', where the need for real-time realistic computer graphics is expressed, and also with recent LINK calls on broadcast multimedia, where the need for such fundamental computer graphics research is called for.

2. Background

There are a number of antecedents to the new method. There is not space to review these fully, and an excellent recent survey can be found in (Watt, 1998). It is assumed that the reader is familiar with this material, and reference can also be made to the attached paper (Slater, 1998). First, what is the scope and goal? The computer graphics convention is to assume a particle model of light, represented as straight-line rays in non-participating media. For simplicity of discussion it is assumed here that the scene is comprised of objects in a vacuum, and objects have material properties, in the manner that they reflect light, represented by a Bi-directional Reflectance Distribution Function (BRDF). Particular idealisations of this are perfectly diffuse reflectors and transmitters, and perfectly specular reflectors and transmitters. Objects can be any continuous geometric shape (discussion of fractal objects is excluded here), and any object can be a light emitter.

Light is emitted from the light sources, bounce from object to object in the scene, and eventually reach the eyes of an observer. Any light path can be represented by a notation introduced by Heckbert (1990). The symbol L represents a light source, E represents an 'eye', S represents a specular surface, and D a diffuse surface. Then a general light path may be represented as $L(SID)^*E$, where ' I ' means 'or', and '*' is arbitrary repetition including no repetition. Hence any light path starts at an emitter, may be reflected from any number of specular or diffuse objects, and may eventually reach the eye. The goal is to be able to illuminate a virtual scene based on any combination of lights, with any combination of light paths, and any set of moving observers. Moreover, objects in the scene may change (for example, may move), thus changing the structure of all light paths in the scene (of course, there would be an infinitely uncountable set of paths!). A further goal is to be able to do this in real-time. This means that an observer should be able to move through the scene, and see correct lighting for each new frame computed at a frame-rate of roughly 20 frames per second or more. As objects themselves move, the lighting should remain correct from frame to frame. The notion of 'real-time' can be relaxed – instead '*in principle real time*' means that each frame should be accomplished in constant time, and a constant time small enough that makes 'real-time' look feasible with a powerful enough computer available today or in the foreseeable future (say, five years).

There is no solution to this set of problems available today.

Computer graphics developed a 'local illumination model' which treats only light paths $L(DIS)E$, where a light-source is a point. Given any point on a surface a diffuse and specular contribution to the reflection of light from that surface may be simply computed taking into account the position of the eye (specular only) and the light source. When surfaces are represented as planar polygons, light 'intensity' is computed only at the vertices, and then interpolated across the polygon – in integrated hardware that similarly deals with z-depth computation. It is this type of system available for most workstations and PC's today, capable of demonstrating real-time performance for certain types of scene. Such systems repeatedly re-render the entire scene database as fast as possible. Dynamic modification to objects is achieved by default, because if an object's geometry is transformed, then in the next frame it will be shown in its correct place, since the graphics rendering pipeline starts from the transformation of objects typically expressed in a local coordinate system for each object.

It is important to note that even today's graphics hardware, although producing real-time solutions for extremely complex scenes, is in fact not '*in principle real-time*'. This is because the rendering time is a function of the number of

objects, ultimately polygons, in the scene. Given any standard graphics hardware, no matter how fast, it is possible to construct a scene complex enough in numbers of polygons, and visibility relationships between them, that it will not perform in real-time.

Ray tracing was the first attempt to achieve global illumination in computer graphics, for ideal specularly reflecting and transmitting surfaces (Whitted, 1980). Ray tracing follows light-paths 'backwards' from the eye, through an image plane, into the scene. There is a primary ray for each pixel on the image plane, and secondary (reflected, transmitted rays) are spawned for each intersection of a ray with an object in the scene. This results in a simple recursive ray-tracing tree. Local illumination is achieved through 'shadow feeler' rays from each intersection point to the (typically point-) light sources in the scene, and therefore shadow umbrae are simulated (but not penumbrae). The argument for such 'backwards' ray tracing, is that most of the light in the scene will never reach the observer. Therefore it is not only impractical, but inefficient to follow light paths originating from emitters. Diffusely reflecting surfaces are not treated correctly – a simple formula based on Lambert's Cosine Law is used at each intersection point to take diffuse reflection into account. Therefore this type of ray tracing can simulate light paths of the form $LS * E$, or $LDS * E$, but cannot take into account the interaction of light between diffuse and specular reflectors or between diffuse and diffuse reflectors. Ray tracing is an inherently 'view dependent' methodology – the eye of the observer is directly taken into account in the light computation, and therefore when the observer moves, the entire ray tracing must be recomputed. There has been some attempt, though without a great deal of success, to capture isomorphisms between ray tracing trees from different viewpoints and directions (Chapman, Calvert, Dill, 1991; Teller, Bala, Dorsey, 1996). The vast majority of work on ray tracing has been to find ways to reduce the number of intersection tests, which completely dominate the computation time (Clark, 1976; Rohlf and Helman, 1994; Fujimoto, Tanaka, Iwata, 1986; Cleary and Wyvil, 1988; Glassner, 1984).

Radiosity is a second major type of method to achieve global illumination. It is a kind of 'antithesis' to ray tracing, in that it models the diffuse inter-reflections of light, but cannot model specular reflection at all (Goral, 1984). In other words it deals only with $LD * E$ light paths. Radiosity is light energy per unit area per unit time. The radiosity equation expresses the radiosity at a small surface area (a patch) as the sum of the energy emitted from that patch, plus the sum of all energy reflected from other patches that reaches this patch. Across all patches in the scene, this results in a set of simultaneous equations. In the initial approach this equation was directly stored and solved, and in later approaches (Cohen, Greenberg, 1985; Cohen, Shenchang, Wallace, Greenberg, 1988; Hanrahan, 1991) is solved iteratively and such that only one row of the equation need be in memory at any point in time. The vast amount of time in radiosity is concerned with form-factor computation, for each pair of patches, of the proportion of energy emitted by one that is received by the other. In practice the scene must be subdivided into a very large number of surface elements, and radiosity is computed at the vertices of these elements. Then interpolated Gouraud shading is used to render the scene. Since radiosity only works with diffuse surfaces; it is clearly 'view independent', so that one radiosity solution supports all possible viewpoints. Thus real-time walkthrough of a virtual scene is possible, using standard graphics hardware. However, if an object moves the radiosity solution is invalid (though there has been some work on incremental repair to the radiosity solution (Chen, 1990)).

Kajiya (1986) introduced the third major method for global illumination, Monte Carlo path tracing. Kajiya showed how in principle the problem of computer graphics rendering could be posed as the solution of a 'rendering equation', an integral equation for $L(x,y)$, the intensity of light along any ray from surface point y to surface point x . The equation involves L itself on the right hand side, since the light energy from y to x clearly also involves the light energy that is incident on y from all other surface points z . Solution of this equation provides a complete solution for rendering, since clearly if $L(x,y)$ were completely specified then results could be found for all directions y to x through a particular image plane onto a centre of projection representing an eye. Kajiya showed how particular approximations to the solution of this equation give rise to the traditional local illumination model, or to ray tracing, or to radiosity. Kajiya proposed a Monte Carlo solution, called path tracing. Path tracing starts with rays from the eye through the image plane into the scene. However, at each intersection point only one further ray is sampled, with probability distribution dependent on the surface reflectance properties of the object in question. Since the whole scheme is based on sampling of rays, non-point light sources can be handled, and also diffuse reflectors (an object being a diffuse reflector simply determines the probability distribution of rays that might be reflected from it). Many rays are fired through each pixel with the colour of the pixel set, for example, as the average of all such rays. Monte Carlo path tracing can therefore achieve $L(SID) * E$ solutions. However, clearly, the method is view-dependent and far from being real-time in principle. Significant advances were made by Ward (1994), whose RADIANCE system is in wide use as an accurate lighting simulation system.

The fourth major method for global illumination is two-pass photon tracing (Jensen and Christensen, 1995; Shirley et al., 1995). This starts by generating a large number of rays at random through the light emitters. The paths are traced until a diffuse reflector is encountered. Diffusely reflecting surfaces are divided into a large number of surface elements. When a photon strikes a diffuse surface, its energy is accumulated into the corresponding surface element. The second pass is a eye based 'backwards' ray tracing computation, where the path is terminated when it reaches a diffuse surface, and uses the energy stored at the corresponding surface element. This type of photon tracing may simulate light paths of the form $LS * DS * E$. A recent example of this method (Walter, Hubbard, Shirley, Greenberg, 1997) properly deals with

the general L(SID)*E paths in its fundamental computation, but is unable to take this into account in its rendering phase which relies on Gouraud shading, so that specularly reflecting surfaces are not shown correctly.

The final approach to rendering to be considered is completely different to those above, and is an example of a technique that has become popular recently, namely 'image based rendering'. This approach does solve one of the major problems – that is, it breaks the dependence of rendering time on the number of objects in the scene. The particular approach of interest here is called 'light field rendering' or the 'lumigraph', introduced independently by two different teams of researchers (Levoy and Hanrahan, 1996; Gortler et al., 1996). The fundamental idea is to construct a discrete representation of all light rays in an enclosed scene. The particular discretisation method used was the 'two plane parameterisation'. A rectangular grid is imposed on each of two parallel planes, and each grid-point on one plane is connected by a straight line to every grid-point on the other plane. With three orthogonally rotated copies of this two-plane arrangement, all possible light rays in an enclosed scene can be approximated. The method then involves estimating the radiance along each ray by taking each grid-point on one plane as centre of projection, and finding all projected images of the scene with view-volume determined by a rectangular region on another plane. All rays then potentially have an associated radiance. Now given any new set of viewing parameters, all the rays that would be involved in forming the corresponding image can be used, and interpolated to form the image. The viewpoint cannot be at any arbitrary point, but must be outside of the convex hull around the scene (otherwise the radiance changes abruptly along the ray). This is best used for constructing synthetic images from real images of real scenes – a very large number of digitised images must be collected under stringent viewing conditions, and then may be interpolated to form completely new images from different viewing orientations. However, images of synthetic scenes can also be produced; clearly though if the synthetic scene is to be photo-realistic, many weeks of processing would be required to produce the large number of images on which the method is based.

This brief survey of global illumination methods has of necessity left out a lot of interesting and useful developments. In particular discrete ray tracing (Yagel, Cohen, Kaufman, 1992) has some distant similarities with the new method to be discussed, but there is not the space to discuss this here. The interested reader is referred to the original papers, or to (Slater, 1998) to see this discussed in the context of the virtual light field.

3. The Virtual Light Field

The 'in principle real-time' solution to the problem of global illumination will seem quite simple, and it is outlined below.

Rendering

Given an enclosed scene any ray in that scene can be represented as a point in 4-dimensional space. In general, for example, a sphere can be placed around the scene, and any ray will intersect the sphere at two distinct points. The spherical angles θ and ϕ at each intersection point could then be used to represent the ray. Hence the space of 'all possible rays in the scene' could be represented as points in R^4 , where R may be restricted to a particular interval of the real line valid for the representation.

This representation is called here the 'virtual light field' (VLF) since it initially represents rays that are potential or 'virtual' carriers of light. Each ray in the VLF initially carries no light energy.

Now an object can be 'rendered' into the VLF. This involves finding every ray that intersects the object, and dividing it into intervals based on the intersection points. Each successive interval along the ray, intersected by a single object, will be either outside or inside the object. For example, if the object is convex, it will split the ray into three intervals – an outside one, followed by an inside one, followed by an outside one. The ray can be parameterised as $p(t) = p + t(q-p)$ where p is the ray origin, q is its terminal, and $0 \leq t \leq 1$. Hence a ray that intersects the object has an associated vector of intervals $[0, t_1, t_2, \dots, t_n, 1]$ where the t_i are the parametric intersection points.

Clearly every object in the scene can be rendered into the VLF in this way. Hence initially, associated with every 4D point in the VLF there is an interval $[0, 1]$, but as the objects are rendered into the VLF, the corresponding intervals are updated.

In fact, each intersection point is not simply represented by its corresponding t -value, but other information is stored as well. The intersection corresponds to a record, called a T-Intersection, containing the following fields: an identifier for the object intersected, the t -value, a direction, a radiance, and an unshot radiance. The radiance values are initialised to zero. The direction allows for energy to be sent in both directions along the ray. One direction is labelled arbitrarily as 'right' (from p to q) and the other as 'left'. Given any intersection, the direction is recorded as 'right' if the normal to the object at the intersection point makes an acute angle with the vector $q-p$, else 'left'.

Hence associated with each ray (or point in 4-dimensional space) is a vector of T-Intersections, ordered in ascending t -value along the ray. A ray may be considered as a 'look-up' index into the table of vectors of T-Intersections.

Propagating Light

So far the light field is still virtual, there is no light in the scene. Objects in the VLF may emit, reflect or transmit light, with any kind of surface material property. Suppose a light is now 'switched on'. Find all rays through this emitter. For any ray identified as intersecting the emitter, the interval, or pair of T-Intersection records, containing the intersection along which light is to be directed can be found, and the identifier of the object at the other end of the interval (if any) is

easily available from the T-Intersection at that end of the interval. When an object is found in this way, it is marked as being in receipt of 'unshot energy'. The energy is written into the radiance and unshot radiance, for the appropriate direction of the T-Intersection with origin at the emitter. This is repeated for each emitter in the scene, and completes the first cycle of the iteration.

Now each object that is in receipt of unshot energy is considered in turn. Take any such object (O). For each ray through the object, find the corresponding T-Intersection for that ray with the object. Suppose that this T-Intersection has unshot radiance being carried into the object. From the normal at the intersection point, and the material properties of the object (its BRDF) a set of rays can be found that will reflect this incoming unshot energy. For example, if the object were an ideal specular reflector then the reflected ray can be found, the appropriate outwards pointing T-Intersection of this reflected ray with the object can be found, and the incoming unshot radiance propagated outwards along this reflected ray (and set to zero on the incident ray). The object identifier at the other end of the reflected ray interval can be looked up, and marked as in receipt of unshot energy. If the object O is an ideal diffuse reflector, then all rays through the intersection point can be found, and treated similarly – with the corresponding energy propagated along them, and the receiving objects marked as in receipt of unshot energy. When all rays with incoming unshot energy to O have been treated in this way, the next object is considered, and treated similarly.

The second cycle is complete when all objects have been treated in this way. Then subsequent cycles through all objects with unshot energy are carried out, and in each cycle, objects are selected in descending order of total unshot energy (so that the most important objects in each cycle are considered first). Unshot energy is only propagated provided that it is above a given tolerance. The cycles continue until there is no unshot energy left anywhere in the system, which is now in equilibrium.

Implementation

The above will seem familiar and unremarkable, for it is essentially just a *description* of how light propagates outwards from light sources through the scene. In order for this to be the basis of a method there has to be a way of implementing it so that it executes in reasonably finite time on a real computer. The basis for accomplishing this is through a discrete representation of the VLF. Suppose that there is such a discrete representation. This must have certain properties. First, the representation should be parameterised according to its 'size' or density of rays. Suppose it is parameterised by a variable n , representing the size of the discrete representation (for example, n could be related to the number of rays). Let r be any arbitrary scene ray. Suppose R_n is the discrete representation of the VLF. For any reasonable measurement of error E_r , and any given error $\epsilon > 0$, it must be possible to choose n large enough so that a ray $r_n \in R_n$ can be found with $E_r(r_n - r) < \epsilon$. Second, it must be possible to find the appropriate $r_n \in R_n$, which is an approximation to r , in constant time. Third, there must be a constant time lookup from any r_n to its associated T-Intersection vector.

Now the description in the previous section is essentially unchanged, but becomes a computable method. The set of all T-Intersection vectors corresponding to rays in R_n could be represented as a 4-dimensional array. The T-Intersection vectors could be represented as binary interval trees. The 'set of all rays through an object' is computable.

Suppose an object is a perfectly specular reflector. When an incident ray strikes the object, the ideal direction of specular reflection r can be computed. However, r itself is not used, but its approximation $r_n \in R_n$ used instead. Similarly, if in the other extreme, the object is a perfectly diffuse reflector, then all rays in R_n through the intersection point can be computed, and energy propagated in the appropriate directions along those. (Of course, in the discrete representation there may be no rays exactly through the intersection point, but the closest set of rays through the intersection point would be used instead).

It is clear that such representations R^n of the VLF are possible. The two-plane parameterisation, for example, would be suitable. However, in practice an alternative representation is used, described in (Slater, 1998). The important point about the representation of R^n is that it should be decomposable into 'parallel sub-fields'. A parallel sub-field is such that all rays within it are parallel. Now if this is the case then only one canonical case is required – that is, when all rays are vertical, through the x-y plane, parallel to the z-axis. It is easy to show that in this case the problem of rendering an object into the VLF is extremely simple. If the object is a polygon or polyhedra, then it can be rendered into a vertical parallel subfield by using a 2-dimensional polygon fill algorithm, with z-interpolation to find the intersection points, identical to raster scanning a polygon in 2D but also computing the z-values. (Unlike a z-buffer, however, all z-intersections with the object are needed, not just the closest intersections). If this can be done for a vertical parallel sub-field, then any other parallel sub-field is just a rotation of the vertical, and hence can make use of this solution. Finding all rays (approximately) through a point is also particularly simple for the parallel sub-field representation. If an object is any arbitrary shape, then all rays through the bounding box of the object can be trivially found in the vertical sub-field case, and then each such ray tested against the actual object. Hence any object that admits a solution to ray intersection can be used as an object in this method.

Avoiding Combinatorial Explosion for Diffuse Objects

Although the method described above is clearly implementable as described, for a scene with several diffuse reflectors there would be an enormous combinatorial explosion. When an incident ray hits a diffuse surface, all rays – approximately – through the intersection point must be followed, and any of these may hit other diffuse surfaces, and so

on. This combinatorial explosion can be completely avoided borrowing an idea from particle tracing. In fact a diffuse surface can be handled by only two passes in each cycle, where a pass means 'find all rays through the object'. For explanatory purposes only, suppose that the object has been sub-divided into small surface elements. Each element will maintain an accumulated energy carried by all rays incident to the surface in any one cycle. A 'gathering' phase finds all rays through the object, and accumulates the unshot radiance into the surface elements. Then a 'shooting' phase finds all rays through the object, and propagates out the accumulated energy found at the surface elements (which are reinitialised to zero at the end of this phase). However, rather than sub-divide objects into surface elements, which is somehow 'alien' to the light-centred philosophy of the VLF method, the energy is accumulated in an additional field of the T-Intersection corresponding to the normal of the object at the intersection point. The intersection-point normal corresponds to a unique T-Intersection for an incident or reflected ray. This has the advantage that the discretisation of the object corresponds exactly to that of the VLF.

Dynamic Modifications to Objects

As will be seen in the next section, the method supports in principle real-time walkthrough of a scene. Unlike most other methods for global illumination (other than radiosity) it also allows objects to be changed without having to recompute the entire set of VLF intersections and lighting. Dynamic modifications require that an object can be deleted from the VLF and an object can be added (a transformed version of the deleted object in this case). Clearly, an object can be deleted or added trivially in the sense of updating the interval trees representing the T-Intersection vectors. However, this would not remove or add the lighting effects caused by the objects. As shown in the attached report (Slater, 1998) negative energy can be propagated through the VLF, corresponding to the radiance that must be subtracted (and added) in the case when an object is added or deleted to the VLF. This will not be an 'in principle real-time' operation, since it depends on the geometric and radiant properties of the object (compare a light emitter with an object that is only a specular reflector for example), but it is useful that this can be done at all.

Viewing and Display

It may have escaped the reader's notice that there is as yet no mechanism described for viewing and actually rendering anything onto a workstation display. This is deliberate, because viewing is actually incidental to the method. Nevertheless, it is obviously necessary in order for human observers to be able to see anything in a sensible way from the VLF. A virtual 'eye' can be placed anywhere in the scene at any viewing orientation. The eye consists of a lens (or any arrangement of lenses) and an image plane. Consider the simplest case of a single convex thin lens. Clearly the lens is just a planar polygon, and all rays through the lens can be trivially and rapidly found. The T-Intersection corresponding to any ray striking the lens can be looked up. If the T-Intersection carries radiance in the appropriate direction (to inside the ray) then the lens formula may be used to compute the refracted ray, which can then be trivially followed to the image plane. Thus radiance is carried to the image plane. The image plane can be mapped to a 2-dimensional window on workstation display, itself divided into 'super-pixels', and any sort of filtering (such as simple averaging of all the rays that hit a particular super-pixel) required at this last stage can be carried out as desired.

There are several important features of this method. First, obviously the lens properties (focal length etc.) can be altered as desired at any time at no cost. Second, it would be easy to apply a filter function at the last stage, for example, to achieve various effects such as anti-aliasing. Third, and most important, the time for this whole operation of capturing an image onto the image plane through a lens is essentially constant for any eye position anywhere in the scene. It is independent of the original number of objects in the scene. It depends only on the size of the discrete VLF representation. Since this is a constant time operation for any particular realisation of the VLF, the claim that this is an in principle real-time walkthrough method is justified. Moreover, it is important to emphasise, this is the first ever method capable of showing different views of globally illuminated specularly reflecting surfaces in constant time – without parallel hardware.

Remarks

As remarked earlier, this is a different way of thinking about computer graphics. Although it naturally contains elements of many previous methods, in combination it is quite different to all of them. It is a light-centred paradigm rather than an object-centred one. It represents the light in a scene rather than displaying the objects. It is strange because there is no traditional rendering pipeline. In many ways it is actually simpler than the traditional rendering pipeline, and represents a direct unfolding of Kajiya's rendering equation. The number of objects does not actually matter as far as rendering to the display is concerned. There is no problem of visibility computation no matter what the complexity of the scene – visibility culling is irrelevant. There is no problem of only dealing with certain types of surface reflectance properties – it deals with all combinations of surface material properties without any special considerations. Many of the problems of traditional computer graphics simply disappear in this different way of looking at things. Of course, it has problems of its own, such as a large memory requirement (1GB memory is estimated for a reasonably sized VLF), problems of scale (the same VLF to represent an entire building will not be suitable for representing a tea-spoon within that building), and problems caused by the nature of its approximations (true ray directions are not used, only approximate ray directions).

The method has had a first implementation by the author. Some results are discussed in the accompanying paper. These early results suggest beyond doubt that the method 'works', and represents a very exciting move forward for computer graphics and virtual reality.

4. Research Goals

There are several high-level interrelated objectives to the proposed research:

- Carry out further research into the foundations of the method and its efficient implementation. Explore the capabilities of the method, calibrate it on a large number of different scenes and surface material types, comparing its results in terms of lighting with other methods.
- Build a new research group that has research and development of the VLF method as its primary goal.
- Obtain significant industrial collaboration for the development of a new graphics architecture based on the method.
- Exploit the method in a number of practical (and commercial) applications.
- Use the method to help stimulate research and development of computer graphics and virtual environment research in the UK (there have been no SIGGRAPH papers originating in the UK for at least 10 years - some UK researchers have been co-authors, but even this is extremely rare).
- Potential shifts in paradigms typically cause problems within any discipline - a significant amount of time will be necessary arguing for the method, putting right the inevitable 'misunderstandings', and publicising the method in popular as well as scientific journals.

A stable and demonstrable implementation will be produced (this goal is very close to being realised in September 1998) as the basis of further research and development. A standard EPSRC research grant will be applied for, which will have two goals: Further understanding of the method itself, including a characterisation of error, and the development of a full-scale implementation that can, for example, take any VRML file and produce a VLF. This will be in conjunction with a specific application area of strategic importance, such as the Broadcast Multimedia program and the Digital-VCE. It would be ideal if this standard grant were to start near the beginning of the Fellowship period itself.

It is not possible to produce a year-by-year account of what will be done in which year. Each of the following needs to be done, and as soon as possible. Ideally they would all be started in the first year, and then unfold over the years following. It is impossible to predict exactly where this may lead.

There will be an *investigation of alternative hardware configurations*. As part of this, funding will be sought to develop a parallel implementation of the method. Some ideas concerning this are outlined in the accompanying paper. Clearly, the 'parallel sub-field' is ideal for exploitation on a parallel hardware configuration, but there will be several other approaches worth considering. At the same time discussions will be opened up with potential industrial collaborators. Since the parallel sub-field idea admits simple 2D based rendering of objects into the VLF, existing hardware designs could be modified for this purpose. There is also the significant problem of a rapid lossless compression scheme for the internal representation of the VLF itself, and operations upon it.

There are many potential applications of the method, outlined briefly:

- The use of the method in a head-mounted display. This is particularly interesting for two reasons. First, the method is obviously ideal for rapidly producing the two (left-eye, right-eye) views for the purposes of stereo display. Of greater interest is the idea that since the method can vary the optical properties of the virtual lenses, it might be advantageous to use this for the purposes of avoiding the accommodation/convergence conflict reported in the literature.
- There are obvious applications in the context of architectural lighting and walkthrough and production of images for cinematic animations.
- Since the method can model geometric lens properties without any special cost, it could be used in technical rehearsal for film and TV production. In other words, not just the lighting, but also the optical characteristics of images seen through cameras could be simulated.
- Since on projection through the lens the z-depth information is obviously available from the T-Intersection, the method can be combined with traditional z-buffering techniques for the purposes of animation of individual objects within the scene. The general issue of how to use the new method to complement existing graphics, rather than just attempt to replace it, is a significant issue.
- If this method fully works out to expectation, for the first time it will be possible to show highly photo-realistic displays in a virtual reality setting. In line with my other area of research (on presence in VEs) there would be strong interest in investigating the impact of this.

In the longer run I have two further ideas that I would like to explore. I am interested to know whether a physical VLF could be built, i.e., one that people could stand in and look around. This would require joint working with physicists, in particular with experts in laser technology. I have some ideas about simulating this situation in virtual reality.

The second idea is a completely new type of graphics architecture, where the VLF plays a central role, and pulses of energy are sent through the VLF from the light emitters directly simulating the 'Propagating Light' phase outlined above. In other words, these energy pulses would be equivalent to frame cycles on current graphics hardware. It is likely that towards the end of the five year period this type of approach would be feasible.

Chapman, J. Calvert, T.W., Dill, J. (1991) Spatio-Temporal Coherence in Ray Tracing, Graphics Interface '91, 101-108.

Chen, E. (1990) Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System, Computer Graphics 24(4), 135-144.

Clark, J.H. (1976) Hierarchical Geometric Models for visible surface algorithms, Communications of the ACM, 19(10), 547-554.

Cleary, J.G., Wyvill, G. (1988) Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision, The Visual Computer, 4, 65-83.

Cohen, M.F., Greenberg, D.P. (1985) The Hemi-Cube: A Radiosity Solution for Complex Environments, Computer Graphics 19(3), 31-40.

Cohen, M.F., Shenchang, Ec., Wallace, J.R. and Greenberg, D.P. (1988) A Progressive Refinement Approach to Fast Radiosity Image Generation, Computer Graphics 22(4), 75-84.

Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F. (1990) Computer Graphics: Principles and Practice, Addison-Wesley, Reading M.A., 2nd ed..

Fujimoto, A., Tanaka, T., Iwata, K. (1986) ARTS: Accelerated Ray-Tracing System, IEEE CG&A 6(4), 16-26.

Glassner, A.S. (1984) Space Subdivision for Fast Ray Tracing, IEEE Computer Graphics and Applications, 4(10), 15-22.

Goral, C.M., Torrance, K.E., Greenburg, D.P. (1984) Modeling the Interaction of Light Between Diffuse Surfaces, Computer Graphics (SIGGRAPH) 18(3), 213-222.

Gortler, S., Grzeszczuk, R., Szeliski, R., Cohen, M. (1996) The Lumigraph, Computer Graphics (SIGGRAPH), Annual Conference Series, 43-52.

Hanrahan, P. et. al. (1991) A Rapid Hierarchical Radiosity Algorithm, Computer Graphics (SIGGRAPH) 25(4) 197-206.

Heckbert, P.S. (1990) Adaptive Radiosity Textures for Bidirectional Ray Tracing, Computer Graphics (SIGGRAPH) 24, 145-154.

Jensen, H.W. and Christensen, N.J. (1995) Computers and Graphics, 19(2), 215-224

Kajiya, J.T. (1986) The Rendering Equation, Computer Graphics (SIGGRAPH), 20(4), 143-150.

Levoy M, Hanrahan, P. (1996) Light Field Rendering, Computer Graphics (SIGGRAPH), Annual Conference Series, 31-42.

Rohlf, J. and Helman, J. (1994) IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics, Computer Graphics (SIGGRAPH Proceedings), 381-394.

Shirley, P., Wade, B., Zareski, D., Hubbard, P., Walter, B., Greenberg, D.P. (1995) Global Illumination via Density Estimation, Proceedings of the Sixth Eurographics Workshop on Rendering, June, 1187-199.

Slater, M. (1998) The Virtual Light Field Global Illumination in Computer Graphics: supporting constant-time walkthrough, technical note forming basis of patent claim.

Teller, S., Bala, K., Dorsey, J. (1996) Conservative Radiance Interpolants for Ray Tracing, Rendering Techniques '96, Proceedings of the Eurographics Workshop in Porto, Portugal, June 17-19, X. Pueyo and P. Schroder (eds), Springer, 257-268..

Walter, B., Hubbard, P.M., Shirley, P., Greenberg, D.P. (1997) Global Illumination Using Local Linear Density Estimation, ACM Transactions on Graphics, 16(3), 217-259.

Ward, G.J. (1994) The RADIANCE Lighting Simulation and Rendering System, Computer Graphics (SIGGRAPH), 459-471.

Watt, A. (1998) *The Computer Image*, Addison Wesley Longman Ltd.

Whitted, T. (1980) An Improved Illumination Model for Shaded Display, *Comm. ACM* 24(6), 343-349, June.

Yagel, R., Cohen, D., Kaufman, A. (1992) Discrete Ray Tracing, *IEEE Computer Graphics and Applications*, 12(5), 19-28.

Supervised Work

Vassilev, T. I. (1996) Fair Interpolation and Approximation of B-splines by Energy Minimisation and Points Insertion. *Computer-Aided Design*, Vol. 28, No. 9, pp. 753-760, 1996.

Vassilev, T. I. (1997) Interactive Sculpting with Deformable Nonuniform B-splines, *Computer Graphics Forum*, Vol. 16, No. 4, pp. 191-199, 1997.

The Virtual Light Field

Global Illumination in Computer Graphics supporting constant-time walkthrough

1. Introduction

A fundamental goal of Computer Graphics research is the achievement of global illumination, resulting in photo-realistic images, at real-time frame rates. Computer graphics is essentially divided into two main approaches: images of a 3D scene can be rendered in real-time, for example, as is required for Virtual Reality, provided that the lighting model used is based only on local illumination. This means that light sources are represented as single points in space, are abstract in the sense that they are not part of the scene but only affect how other objects are illuminated, and only the direct illumination from light source to object is considered, there being no reflection of light between objects. Moreover, the actual lighting model used is only loosely based on the real physics of light transport. The scene must ultimately be decomposed into planar polygons, usually triangles.

An alternative paradigm in computer graphics is to produce scenes that are photo-realistic in some sense, based on a global illumination model, where light transport is not just between light sources (emitters) and non-emitting (but reflecting) objects, but between the objects themselves. Ray tracing provides a method that models scenes with ideal specular surfaces, but does not correctly deal with diffuse surfaces. Each time that the viewpoint is moved the entire ray tracing algorithm has to be executed again, which can take several hours for each image. Radiosity is another paradigm that is based on a physical model of light transport between ideal diffuse reflectors - although it is view-independent, it cannot at all handle specularly reflecting objects.

There are Monte Carlo methods that attempt to model light transport based on probabilistic sampling of light rays for scenes consisting of both diffuse and specularly reflecting surfaces. Monte-Carlo path tracing, although producing highly realistic images, is view-dependent - the entire image must be recomputed with every change in view. Photon-tracing methods, however, follow large collections of photons emitted probabilistically from the light sources and trace them through the scene until absorption.

¹ This paper has formed the basis of a UK patent filing, and should be treated in the strictest of confidence.

When a photon strikes an object, this information is retained with the object, and then used in a later rendering process. Photon tracing again cannot typically handle all interactions of light between different types of reflecting surfaces.

Even in the real-time local illumination approach there are increasing problems with the standard graphics pipeline: it is based on polygons, there are millions of them for a realistic scene, visibility computation is a major problem to the extent that even for relatively small scenes with visual complexity state-of-the-art graphics hardware cannot achieve real-time. The linear dependence of rendering time on numbers of polygons and non-linear dependence on scene depth complexity is a major problem, with many ingenious but inelegant solutions, which often only apply to scenes of a particular topology, or to particular view-paths through the scene.

This paper introduces a new approach to computer graphics. This integrates real-time and photo-realism, and produces rendering times that are independent of the number of objects in the scene and their depth complexity. It properly models light transport in scenes with diffusely and specularly reflecting surfaces. It does not require any important recomputation with a change in viewpoint, but rather can rapidly generate a new image from any viewpoint. Its rendering speed does not depend on the number of objects. Polygons play no special role in this at all. It does not rely at all on the current graphics pipeline. Many current preoccupations such as visibility culling and level of detail become non-issues in this new paradigm.

The new approach represents a kind of ontological break with the standard paradigm. The starting point for computer graphics seems to be empty space which then has objects and light sources added to it. Objects not only contain information about their geometry and material properties, which govern how they reflect light, but also are used to store their surface radiance, the actual light energy they reflect. Objects are rendered onto the image plane by a complex sequence of operations involving projection (from 3D to 2D) and the determination pixel by pixel on the view-plane of both the visible object 'through' that pixel and the colour at which the pixel should be rendered. Familiarity amongst computer graphics practitioners with the rendering pipeline leads to a collective 'forgetting' about how computationally intensive this process really is. Many years of research and development have hidden the complexity through specialised geometric and pixel based algorithms embedded in hardware. The degree to which this process is 'unnatural' can be appreciated by the fact that in local illumination models and in standard ray tracing, if the viewpoint is aimed towards a light source, the light source itself cannot be seen, even though it is supposed to be the case that objects are lit only because there is light. In fact, even if all the light sources were 'switched off' most other scene objects would still be seen, because of the fiction called 'ambient light' - each object is in fact treated as an emitter which has its own light representing its reflection of all background illumination in the environment.

Computer graphics follows essentially an object-centric paradigm - objects typically store all the information that permits a scene to be rendered. Even in the most recent and sophisticated Monte-Carlo particle tracing system, objects are the ultimate carriers of radiance, are broken down into triangles, and these triangles are rendered through Gouraud Shading interpolation on the display.

This paper introduces a new *light centred* paradigm. It starts with potential (or virtual) light in an environment. Objects are rendered into this potential light field. Some objects emit light, thus transporting photons along a subfield of the virtual light paths, illuminating other objects, which in turn light up ever increasing subfields of the total virtual light field, until all photons are absorbed or lost out of the system and an equilibrium distribution of light energy is reached. The representation of the illuminated scene is now entirely captured within the total virtual light field (the original object specifications could be thrown away). A 'being' can be placed into this total light field, one capable of focusing light onto a surface (for example a planar surface) inside its 'eye' - and we the external human observers essentially see the world through the eye of this being. The virtual eye (or eyes) can be moved around - notice without any recomputation of the illumination, but only refocusing of light into the virtual eyes. This walkthrough produces images at frame rates independently of the original number of objects in the scene (which by now is irrelevant) and only dependent on the resolution of the total light field.

Although the paradigm is new, it has a number of antecedents. We discuss the background in Section 2. The method is described in detail in Section 3, and implementation considerations discussed in Section 4. Unusually for a global illumination method this one can deal with changes to scene geometry (though not in constant time). This is discussed in Section 5. Some results are given in Section 6, and a method for parallelism in Section 7. Section 8 discusses various features of the method, 9 deals with applications, and 10 is the conclusions.

2. Previous Approaches

There are six approaches to global illumination which share some characteristics with the virtual light field approach: ray tracing, ray space methods, Monte-Carlo path tracing, photon tracing, discrete ray tracing, and image rendering based on the light field or lumigraph approach. Of these the latter bears greatest similarity to the approach of this paper, in that a similar data structure can be employed, but as will be seen the methods are quite different.

2.1 Ray Tracing

Ray tracing was introduced by Whitted (1980). Ray tracing requires a pin-hole camera model and point light sources, and a non-participating medium (in fact a vacuum representing an approximation to air). Its fundamental operation is to trace a ray with origin at the centre of projection (COP) through a point on an image plane. The point is a pixel (or 'sub pixel' in the case when some anti-aliasing method is required) - the purpose being to find the 'intensity' for the pixel, by tracing the ray back through the scene, bouncing from object to object, until it goes out of the scene or makes a negligible contribution to the intensity.

In order to achieve this the nearest object intersected by the ray is found (at point P). A bi-directional reflectance (BRDF) function, in the most general case, will determine the subsequent path of the ray. In fact ray tracing classically only follows three specific paths at each object intersection. The first is a 'shadow feeler' ray - the rays from the object

intersection P to the set of point light sources are followed, and each such ray that does not intersect another (opaque) object before reaching the light source, adds a local illumination contribution to the pixel. This intensity is computed as the sum of an 'ambient' term, representing all background light in the scene, and terms for each light source based on Lambert's Law for ideal diffuse reflectors plus a highlighting term. Notice that there is no global illumination for diffuse reflection, but only the local contribution of each light source to P (and therefore to the current pixel). If the object is an ideal specular reflector, then a new ray is computed in the direction of specular reflection, based on the law of angle of incidence equal to angle of reflection, and the reflected and incident rays being in the same plane. The ray tracing function is recursively invoked with this reflected ray, and the result added to the intensity at P. Similarly, if the object is transparent, then using Snell's Law new ray directions through the object and out again can be computed, and again, the ray tracing function recursively invoked to produce an intensity to be added to that for P. This ray tracing function is called for each primary ray with origin at the COP and through each relevant point (pixel or subpixel) on the image plane. The result is an image where specular reflection and transmission have been simulated.

Ray tracing was a major advance in the synthesis of photo-realistic images in computer graphics. It is very computationally intensive - most of the computational effort is involved in ray-object intersection calculations. A naive implementation would need to test each ray against each object for example. Much of the (vast) research on ray tracing has concentrated on speeding up this process by substantially reducing the number of ray-object intersection tests. Such methods have included bounding volumes and hierarchies (Clark, 1976; Rohlf and Helman, 1994) where rays are tested first against simpler objects that encapsulate the actual scene objects. Only if the test against the simple bounding object (an axis aligned box, or a sphere) succeeds is there any need to test the ray against the actual scene object. Such bounding volumes can be organised hierarchically making use of spatial coherence of objects in the scene - so that groups of objects close together can be grouped into a bounding volume, and then groups of bounding volumes put together in a larger volume, and so on. The bounding volumes may be organised as a tree with root enclosing the entire scene, and the objects at the leaves. The ray is filtered down the tree until either it intersects no bounding volume, or else finally intersects an object at the leaves.

Spatial partitioning methods exhaustively partition the space occupied by the scene into smaller regions each of which maintains a list of identifiers of objects that intersect it. Uniform partitioning (Fujimoto, Tanaka, Iwata, 1986; Cleary and Wyvil, 1988) divides the scene space into a regular three-dimensional grid of small cuboids or cells. The path of a ray through this space-subdivision can be quickly computed, and only objects in the cells along the ray path are candidates for intersection. Adaptive partitioning, such as oc-trees (Glassner, 1984) subdivide the space in a way that depends on the distribution of objects - so that more densely occupied regions are subdivided to a greater extent.

Notwithstanding the great deal of research, ray tracing remains a computationally expensive process, with no prospect of real-time interaction on conventional computers. Clearly, with each change in viewpoint the entire ray-tracing algorithm must be carried out again. There have been some efforts to exploit coherence from image to image as the

camera moves (for example Chapman, Calvert, Dill, 1991; Teller, Bala, Dorsey, 1996), but these have not achieved results sufficient for anything approaching real-time walkthrough.

2.2 Ray Space Methods for Ray Tracing

Ray classification schemes provide another type of approach to speeding up ray tracing. The primary space of interest in this case is the space of rays rather than the usual space of objects. The most notable and elegant example of this was introduced by Arvo and Kirk (1987). A ray has an origin (x,y,z) and a direction given by angles (θ,ϕ) and hence may be considered as a point in 5D space. Arvo and Kirk in fact represented all possible rays in the scene as 6 sets of 5D points. An axis aligned bounding box is placed around the scene, and each face of the box has a 2D UV coordinate system. Hence a ray can be represented by its origin (x,y,z) , the side of the bounding box that it intersects (Left, Right, Top, Bottom, Near, Far), and the UV coordinate of intersection. Hence all possible rays can be represented by the six sets of all points (x,y,z,U,V) .

The algorithm exploits ray-coherence - that is, rays that are 'close' to one another are likely to intersect similar sets of objects. In brief as each ray is considered a 32-tree subdivision (a 5D version of an oc-tree) is lazily formed for each of the 6 faces, where at any moment any leaf of the tree represents a bundle of (similar) rays and has a corresponding 'candidate' set of objects - i.e., any object in the candidate set is likely to be intersected by at least one of the rays in the corresponding ray subset.

The important point for this paper is not the details of the algorithm, but the idea that it is the space of 'all possible rays' that is of primary importance rather than the space of objects. The algorithm forms a partition of ray space rather than of object space.

Muller and Winckler (1992) introduced another ray-space method - though for quite different reasons. 'Breadth-first ray tracing' is designed not especially for speed, but to save on the substantial memory requirements that occur when there is a huge object data base. The idea is that rays are kept in main memory and object information may be stored on disc. Each ray is stored as an origin, parameters representing the line equation of the ray, the current intersection point along the ray, and the object corresponding to that intersection point. In a naive implementation of this algorithm, each object is read from disc into memory and intersected against all primary rays. Using a kind of 'z-buffer' all primary rays that intersect anything would point to the nearest object intersected, with, of course, the corresponding intersection point. A similar process is now repeated for all 'shadow feeler' rays. It is then continued for all reflected rays, then transmitted rays, and so on, to any required level of depth. The ray tracing is therefore breadth first - dealing with all rays in each iteration rather than recursively moving through object space. Namamaru and Ohno (1997) improved this approach later by using uniform space subdivision for the rays. In fact this space subdivision was carried out in object space - which was partitioned into a regular grid of cuboid cells, and then each cell maintains a list of pointers to rays that pass through it. In this way, as objects are read from the database, only those rays that pass through the cells occupied by the object need be tested. Although the breadth-first ray tracing avoids recursion and therefore can deal with extremely large databases, it is generally slower than the normal depth first ray tracing,

and, of course, has all the usual properties of ray tracing - dealing only with specular reflection at the global level, and being viewpoint dependent.

2.3 Monte-Carlo Path Tracing

Kajiya (1986) unified a number of computer graphics lighting models into one overall model based on a rendering equation for the intensity of light $I(x,y)$ passing from point y to x , where y and x are constrained to be on surfaces in the scene (or, x may be on the viewplane, receiving light from surfaces in the scene). The resulting integral equation is shown in (1) where the integral is taken over the union of all points on surfaces S , where S includes one bounding surface that encapsulates the whole scene (for example, a surrounding sphere):

$$I(x, y) = g(x, y) \left[\epsilon(x, y) + \int_S \rho(x, y, z) I(y, z) dz \right] \quad (1)$$

$g(x,y)$ is a geometry term which is either 0 if x and y are not visible to one another and $1/r^2$ otherwise, where r is the distance between them. $\epsilon(x, y)$ is the intensity of emitted light from y to x , and $\rho(x, y, z)$ is a dimensionless quantity representing the unoccluded reflectance from z to x via y . Hence the equation states that if x is visible to y , then the intensity of light energy from y to x is given as that directly emitted from y to x (if y is a source) plus the integral, over all surfaces, of light that arrives at y from any other point z that ultimately reaches x . Hence briefly $I(x,y)$ consists of the light emitted from y to x plus the total light reflected from y to x .

If x is considered as a point on the surface of the viewplane, and y considered as the point of intersection with a surface visible from the COP on a light of sight through y , then a solution to (1) for each such x would provide an image accounting for all the relevant interreflection of light through the environment.

Equation (1) can be rewritten in operator form as:

$$I = g(\epsilon + MI) \quad (2)$$

where M is the integral operator, which applied to $h(u,v)$ yields $\int_S \rho(u, v, z) h(v, z) dz$.

Rewriting (2):

$$(1 - gM)I = g\epsilon \quad (3)$$

and provided that M satisfies convergence criteria, always met in this application:

$$I = g\epsilon + gM g\epsilon + (gM)^2 g\epsilon + (gM)^3 g\epsilon + \dots \quad (4)$$

Suppose x is a point on a surface (for example on the viewplane) and y is a point on a surface visible to x . Then the first term represents any direct lighting from y to x . Unless

y is itself a light source this will be zero. It is interesting to note that in classical ray tracing, this term is ignored - if there is a line of site from a point light through to the COP through the viewplane (at x), this will typically not be accounted for in ray tracing - the light sources are invisible!

The second term can be expanded as follows:

$$g(x, y) \int_s \rho(x, y, z) g(y, z) \epsilon(y, z) dz \quad (5)$$

This accounts for light from y to x where y has itself been lit by all light sources visible to it. If the light sources are points then the integral becomes a sum over all visible sources (z) to y. This term accounts for local illumination. However, it is still more sophisticated than the standard graphics pipeline, since (5) does include the computation of shadow umbras (in the case of point sources). As Kajiya points out the standard pipeline leaves out the g term under the integral sign (light source visibility between surfaces is not usually taken into account).

The third term is:

$$\text{let } h(y, z) = g(y, z) \int_s \rho(y, z, w) g(z, w) \epsilon(z, w) dw \text{ in}$$

$$g(x, y) \int_s \rho(x, y, z) h(y, z) \epsilon(y, z) dz \quad (6)$$

The third term now includes one level of inter-reflection between surfaces. x receives light from y that is not just directly caused by emitters, but from other surfaces (z) which do directly receive light from the emitters (w).

So it goes on, with infinite expansion of (4) accounting ultimately for all light energy received at x.

As has been pointed out, the classic rendering pipeline is a special case approximation of this, and similarly (by using only point lights, and choosing a simple form for ρ) ray tracing is another special case. By considering the situation where all surfaces are perfectly diffuse reflectors Kajiya also reduces the rendering equation to the radiosity equation introduced by Goral, Torrance and Greenberg (1984).

Kajiya showed how to solve the rendering equation by (Markov Chain) Monte Carlo integration. (Monte Carlo integration had previously been used by Cook, in his distributed ray tracing approach, though for different purposes (Cook, Porter, Carpenter, 1984)). Kajiya called the method 'path tracing' since it involves following the path of a photon on a random walk from surface to surface - as in ray tracing, a 'backwards' path from the viewpoint into the scene. Consider a ray starting at the viewpoint and through a point (x) on the viewplane, intersecting the first visible surface at point y. Compute random rays to each of the light sources (the distributions determined by the radiant properties of the sources), and note that the sources need not be points. For each such visible ray compute the local contribution of intensity at point y, and add this to the

intensity for x . Depending on the material properties of the surface (in terms of the coefficients of diffuse and specular reflection, and the degree of transparency of the object) choose another ray at random, determine its intersection with the environment (z), and repeat the process, each time adding to the intensity of the pixel. Surfaces may have both diffuse and specular material properties - these determine the distribution of random rays to be fired at each surface intersection. Hence at each ray intersection, rays are always fired to the light sources (to get the local contribution), and then one further ray is fired with probability distribution determined by the material properties of the surface. Unlike standard ray tracing, where at each intersection point all rays (specularly reflecting and transmitting) are followed, here only one such ray is followed. For each pixel (or sub-pixel) a number (N) of such paths are followed, and the final intensity is the average of all of these. Kajiya used $N=40$ for all examples.

The advantages of such Monte Carlo path tracing are clear: all types of surface can be considered, and greater weight is attached to the most important rays (the primary rays, rays to the light sources) without following paths that contribute little to the final intensities. In standard ray tracing, the trees may recurse to enormous depth, but with little contribution of the rays at and near the leaves of the tree.

In terms of producing a real-time solution to the global illumination problem, however, this technique is quite unsuitable - it is still view dependent, with the entire computation required again when the viewpoint moves.

Much of the research following from this approach has been on variance reduction techniques. Each pixel intensity is provided by an estimator based on the ray path samples. By taking into account visibility and the material properties of the surface in path generation there is a considerable degree of importance sampling, a powerful variance reduction technique. There have been considerable advances in this approach (for example, Arvo and Kirk, 1990).

2.4 Photon Tracing

Photon tracing is another Monte Carlo technique though with a different philosophy and outcome. In its most recent form it does provide a potential real-time solution for walkthrough of globally illuminated environments. An example is by Jensen and Christensen (1995) who used a two-pass method in tracing rays from the light sources, combined with the usual method of tracing rays to the viewpoint. Shirley et. al. (1995) introduced a method whereby only paths from the light sources are followed. This has been recently extended by Walter, Hubbard, Shirley and Greenberg (1997) into a multi-stage method which has the advantage of producing a globally illuminated scene for real-time walkthrough, which makes use of statistical density estimation techniques.

The method involves producing a fine triangulation of all (polygonal) objects in the scene. Photons are traced from light sources, with energy determined by the energy distribution of the emitters. Intersections with objects are followed by further Monte Carlo based tracing, until the particles are finally absorbed. When there is a hit on an object surface, the information is stored in the corresponding triangle. At the end of this process (which can take several tens of hours for a realistic scene) all triangles are sorted into object order. Each triangle potentially has an associated distribution of particle hits,

which is then used to estimate radiance density function over that patch. Finally adjacent triangles which are similar enough according to some tolerance criterion are merged together (this last step only to reduce the total number of triangles in the interest of rendering speed). The scene can then be rendered in real-time using Gouraud shading.

The method results in very impressive photo-realistic images - though with a number of drawbacks. Specular reflecting and transmitting surfaces are used correctly in the particle tracing phase, but due to the final Gouraud shading, they are not displayed as such. The method relies on all primitives in the scene being represented as polygons. Darker regions may not receive enough particle hits to illuminate them at all, with brighter regions taking up all the storage space - there is a non-uniform distribution of the emitted light across the scene.

This method is the epitome of the object-centric way of thinking: light energy is captured on the objects, then those objects are displayed (as if the light has never existed). In all these methods it is as if the objects are lit, and then the lights thrown away. In the method to be discussed in Section 3, the objects are fully encapsulated into the light, and the original object definitions could be thrown away rather than the lights.

2.5 Discrete Ray Tracing

Discrete ray tracing belongs to the volume visualisation approach to computer graphics (Kaufman, 1996). It is a quite different approach to ray tracing, introduced by Yagel, Cohen and Kaufman (1992), sharing many of the features to be introduced in the new method. In conventional computer graphics the fundamental entity is the 'pixel', representing a location on a display device, which may be independently set to a colour intensity. In volume rendering the display space is three-dimensional, and the smallest corresponding entity is the voxel, which may contain both colour information and information regarding the (solid) object of which it is a part. In practice voxels are small cuboid, non-overlapping regions, small enough so that at most one object can intersect a voxel.

In DRT objects are first 'rendered' into the voxel space. Each intersected voxel records information regarding the exact normal to the original object, information about the material properties of the object, and also rays are traced, through the voxel space to each of the light sources. For each light source the voxel records a two-bit code to indicate that it is visible, invisible, or visible through another translucent object.

Tracing rays through the voxel space involves finding the 'best' set of voxels along the path of a ray, much like the famous Bresenham or DDA algorithms for finding the best line representations in 2D pixel space.

The rendering phase begins the same way as in conventional ray tracing. A ray is traced through the 2D image plane with origin at the COP. The ray is followed along the voxel path, until it reaches a non-empty voxel. Of course the first one that it finds is the relevant intersection point. The 'shadow feeler' rays are already known (they were pre-computed immediately following the object rendering phase). DRT then follows the standard recursive ray tracing method - of spawning reflected and transmitted rays.

The advantages of DRT over conventional ray tracing are:

- The intersection computations are no longer dominant - since rays are followed until they hit the first non-empty voxel. There is no searching for the nearest intersection point, but the time spent by rays traversing the voxel space is instead significant.
- The ray tracing time is essentially independent of the number of objects. As pointed out by the authors, the greater the number of objects in fact the less time ray tracing, since the rays will hit non-empty voxels sooner in more densely occupied regions of space. Of course the ray tracing time does depend on the 3D resolution of the voxel space, since this determines the path-length for ray traversals.
- Many view-independent attributes are pre-computed - such as the shadow feelers, normals, and textures. Hence a change of view requires significantly less new computation compared to conventional ray tracing.

In spite of these advances, DRT remains a highly view-dependent approach - still the majority of the computation (the construction of the ray tracing tree for each image plane pixel) needs to be carried out for each change in view.

The philosophy of the method remains largely object centric; however, now there is only one type of object that matters, which is the voxel itself.

2.6 Radiosity

The class of techniques known as 'radiosity' (introduced by Goral, 1984) is a rather different approach to global illumination from all those above. It is suitable for scenes where all materials are ideal diffuse reflectors. Almost all methods rely on a polygonal scene geometry, and where each polygon is further subdivided into small patches, which in turn may be further sub-divided into surface elements. The fundamental equation, which may be derived from (1) relates the radiosity, the energy per unit time per unit area of a patch, to all other patches in the environment. For a specific wavelength (omitted from the equation):

$$A_i B_i = A_i E_i + \rho_i \sum_j F_{ji} B_j A_j \quad (7)$$

where B_i is the radiosity per unit area per unit time, A_i is the area, E_i is the radiant energy emitted per unit area per unit time, ρ_i is the diffuse reflection coefficient for the give wavelength, and F_{ji} is the 'form-factor' representing the fraction of energy from surface j that reaches i .

It is possible to eliminate the A s from the equation, and to rearrange the set of equations for the unknowns (B) in terms of the known quantities - and solve the equations. However, the form-factors are unknown, and the major work in radiosity is involved in estimating the form-factors (for example, Cohen, Greenberg, 1985; Cohen, Shenchang, Wallace, Greenberg, 1988; Hanrahan, 1991).

One method in particular is noteworthy here by Wallace, Elmquist, and Haines (1989) who estimated the form-factors through ray casting - a number of rays are cast from a vertex to a patch, and the proportion of hits estimates the differential form-factor from the vertex to the patch area.

The construction of the patch-element mesh is crucial. The ideal is to sub-divide those patches into a finer mesh where there is likely to be a discontinuity in radiosity - in particular at shadow boundaries. Hence often a two-pass solution is adopted, where first a shadow algorithm computes these lines of discontinuity, which form the basis of mesh generation, and then a radiosity algorithm executed.

Radiosity is obviously limited to diffuse environments. Specular effects may be added in a further post-process, but of course, as pointed out by Jensen, and by Arvo, this cannot be used to properly model diffuse-specular interreflections, since each of these phases is carried out independently.

Radiosity does result in the capability for real-time walkthrough - radiosities are computed ultimately at the vertices of the patches, and then Gouraud shading can be employed.

2.7 Light Field Rendering (Lumigraph)

Kajiya's rendering equation used the intensity $I(x,y)$ across all surface points x and y . The light-field, lumigraph, or plenoptic function provides the radiance $L(x,y,z,\theta,\phi)$ at any point (x,y,z) and in direction given by angles (θ,ϕ) . If this function were known for each point and direction, then rendering would be a matter of estimating the radiance at each point on an image plane in the required viewing direction.

Levoy and Hanrahan (1996) and also Gortler et. al. (1996) provided a way to estimate the light-field (or lumigraph) for virtual scenes, and from images of real scenes. The idea is very simple, and we outline only the main features in so far as these are similar to those of the next section.

Radiance is constant along a given direction in 'free space', that is, where there are no discontinuities due to intersections with objects. Hence radiance is constant along each direction outside the convex hull of a scene.

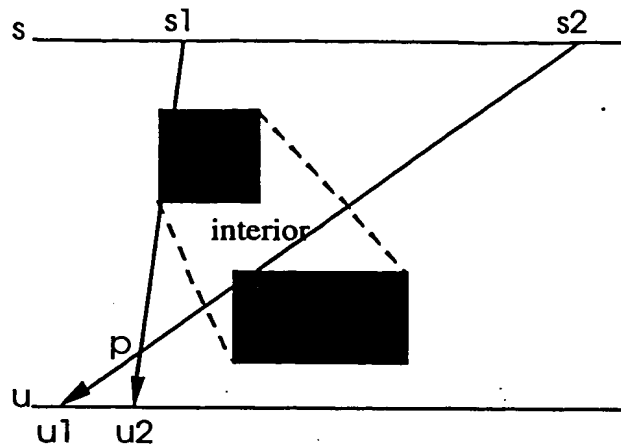


Figure 1
Parameterising the Light Field (2D example)

Figure 1 shows a simple 2D example. Consider any point along line u as the viewpoint, and direction given by a ray with origin along u intersecting line s . All rays intersecting the scene consisting of the two shaded boxes can be parameterised in 2D using (u,s) . However, this parameterisation is only valid for views outside the convex hull of the scene, for it is only for these rays that the radiance is constant. Views from anywhere in the interior would not be possible, since the radiance abruptly changes along such paths.

Given this restriction, a subset of the lumigraph for the scene can be parameterised using (u,s) only, and views outside the convex hull could be constructed from this. For example, suppose an image for a pin-hole camera were required from point p for the view 'volume' within the arrowed lines. Then by capturing the radiance from all the rays between the arrowed lines such an image could be constructed.

In order to construct the complete lumigraph (for the convex hull of the scene) two copies of the 'light slab' defined by u and s would be needed - one as shown, and the other with, for example, two vertical lines on either side of the scene. Nevertheless, the overall representation of the lumigraph is still two-dimensional, albeit with two such two-dimensional representations.

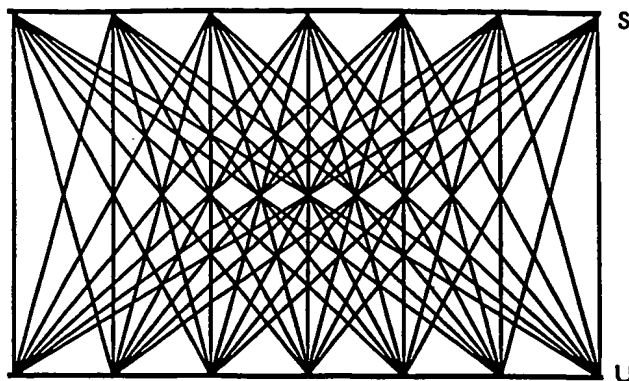


Figure 2
Discrete Representation of the Light Field

Staying with this two-dimensional example, and with the single 'light slab' for u and s , the light field radiance would be a continuous function of u and s , $L(u,s)$. For computational purposes a discrete representation would be needed. The two-line parameterisation is shown in Figure 2. Grids are imposed on the u and s axes, and then all possible lines between each grid point on one to each grid point on the other forms the discrete representation of the set of all such lines.

The light-field would be estimated by choosing each grid point (u_0) on u as a viewpoint for a pin-hole camera with image plane behind u , and an interval (I) of points on s . The intervals partitioning s must be exclusive and exhaustive, and typically the same size. For each such I do the perspective projection through u_0 . If there were N intervals on s then each grid point would form N images. If there were M grid points then there would be $N \times M$ images. In this way, every line that intersects the convex hull of the scene would have an associated radiance determined by the 'rendering' of the image formed by the set of perspective projections.

For 3D scenes the 'two-line' parameterisation is replaced by a 'two-plane' parameterisation, the first parameterised by (u,v) and the second by (s,t) . The light field (in fact a sub-set of it) is then represented by $L(u,v,s,t)$, and discretised by all possible lines between the two planes. The st plane is subdivided into squares, and each projected through each grid-point on the uv plane to form an associated image - and thereby providing radiance values for each (u,v,s,t) combination representing lines that intersect the convex hull of the scene.

A desired image from another viewpoint and viewing parameters can then be constructed by sampling the corresponding set of lines through the viewpoint and in the required directions. This discussion has not included issues of avoiding aliasing through filtering and sampling L for any (u,v,s,t) by interpolation, but these are not required in the sequel to this paper. Suffice it to say that given any image point (x,y) the two papers introducing this method showed how it is possible to use texture map hardware to rapidly recover the required (u,v) and (s,t) values.

Apart from the problem of views restricted to being outside the convex hull of the scene, the light-field method does provide a real-time solution for walk-throughs of scenes with global illumination, though its most important practical contribution is actually for image-based rendering rather than for synthetic scenes, since clearly a method such as photon tracing also provides a solution. In fact some other method of producing photo-realistic images is needed in order to sample the lumigraph - for example each such image being produced by ray tracing. This would be an extremely time consuming task.

The next section shows how to estimate the light field directly, without the need for the production of 2D images. The same discrete representation of the light-field may be employed, but the method of estimating the radiance is quite different, more direct, and conceptually more appropriate to the task.

3. Virtual Light Field (VLF)

The new approach has a number of components: the virtual light field itself, the process of rendering into the virtual light field, illuminating the VLF, and then image capture.

3.1 What is the VLF?

In the two-plane parameterisation let $R(u,v,s,t,d)$ represent the line from point (u,v) on the uv plane to point (s,t) on the st plane. The line is oriented, for example, with $d = \text{'right'}$ the line is from (u,v) to (s,t) and with $d = \text{'left'}$ from (s,t) to (u,v) . Let \mathcal{R} be the set of all such lines, over the valid range of parameter values for u,v,s,t .

The space of all such lines \mathcal{R} represents a '*virtual light field*' in the following sense. In a situation where there are no light emitters, the radiance carried along each such ray will be zero. When a light emitter is inserted into the environment then associated with each ray there will be intervals along the ray carrying radiance. Each ray may be thought of as a *potential* carrier of radiance, a *virtual light ray*.

Suppose an object is inserted into this virtual light field, and that the object is closed, so that any ray intersects it either 0 or an even number of times. Hence each ray that intersects the object will be partitioned into intervals '*inside*' the object, and intervals *outside* the object.

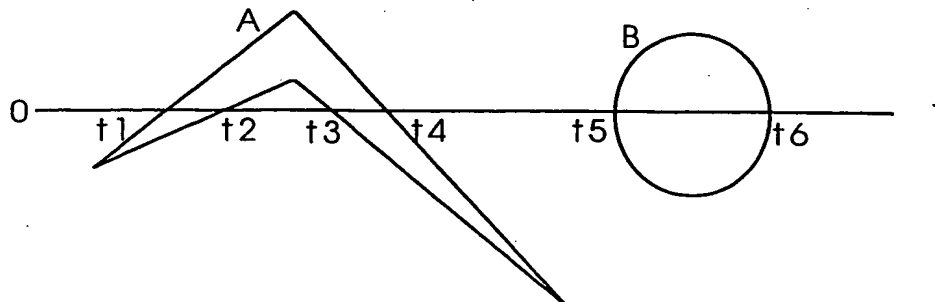


Figure 3
A Virtual Light Ray Intersecting Two Objects

For example, consider the ray shown in Figure 3, parameterised with t , where $t=0$ (corresponding to a point on the uv plane) and $t=1$ (corresponding to a point on the st plane), with the 'right' direction going from 0 to 1. Now object A is inserted, and the ray becomes partitioned into the sequence of segments $[0,t_1]$, (t_1,t_2) , $[t_2,t_3]$, (t_3,t_4) and $[t_4,1]$. The first, third, and fifth segments correspond to those parts of the ray outside the object and the even numbered ones are inside the object. Suppose for the moment that the object were opaque, then all reflected light from the object is carried along those segments that are outside the object. Those segments that are outside the object may be thought of as the radiators of light. The direction parameter indicates the direction along the ray of any potential radiance.

A second object (B) is inserted that also intersects this ray. Then the partition is updated: $[t_4,1]$ is replaced by $[t_4,t_5]$, (t_5,t_6) , $[t_6,1]$. Suppose that object B were a light emitter, then segment $[t_4,t_5]$ would be lit in the 'left' direction and segment $[t_6,1]$ would be lit in the 'right' direction.

For each line $R \in \mathcal{R}$ there is a sequence of parametric t -values $[0,t_1,t_2,\dots,t_n,1]$, corresponding to intervals along the ray that are outside and inside objects. More precisely, each parametric point is associated with the following data structure, referred to as a T-intersection:

$$T = \{t, \text{identifier}, \text{direction}, \text{radiance}\}$$

where t is the parametric intersection point, 'identifier' identifies the object that caused the intersection, direction is either 'right' if the direction of any potential radiance from the object is in the 'right' direction, or else 'left', and 'radiance' is the actual radiance (if any) in the stated direction.

The representation of the ray shown in Figure 3 would be as follows (assuming the radiance as zero):

$$\begin{aligned} T_1 &= \{t_1, A, \text{left}, 0\} \\ T_2 &= \{t_2, A, \text{right}, 0\} \\ T_3 &= \{t_3, A, \text{left}, 0\} \\ T_4 &= \{t_4, A, \text{right}, 0\} \\ T_5 &= \{t_5, B, \text{left}, 0\} \\ T_6 &= \{t_6, B, \text{right}, 0\} \end{aligned}$$

It follows that for every potential ray parameterised by (u,v,s,t) there is a corresponding sequence $T(u,v,s,t)$ of such T-intersections. Let \mathcal{S} be the set of all such T-intersections - more exactly this is what is meant here by the VLF.

3.2 Rendering Objects into the VLF

Initially, before any objects are rendered, \mathcal{S} consists of the set of all $T(u,v,s,t)$, all being sequences representing just one interval, from $t=0$ to $t=1$. The process of 'rendering' an object into the virtual light field updates the sequence of T-intersections for rays that intersect that object. This updating involves simply sub-dividing already existing intervals by the intersection points caused by the object. This is computationally very simple - each sequence of T-intersections for a ray can be represented, for example, by a binary-search tree.

In the above discussion it was assumed that objects were closed and opaque. However, there is no need for this. Suppose A were transparent, then two T-intersections would exist for each intersection point, with opposite directions. If an object were not closed (for example, a single oriented polygon) then a single T-intersection per intersected ray would be inserted.

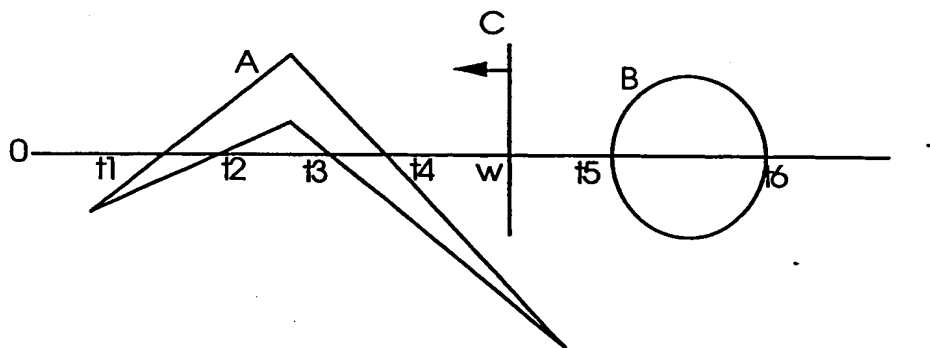


Figure 4

A Virtual Light Ray Intersecting Two Closed Objects and A Polygon

Figure 4 shows what would happen in this circumstance - where C represents a polygon, belonging to an opaque object, inserted between objects A and B. Then the T-intersection sequence would become, if the new intersection point were at w:

```
{t1,A,left,0}
{t2,A,right,0}
{t3,A,left,0}
{t4,A,right,0}
{w,C,left,0}
{t5,B,left,0}
{t6,B,right,0}
```

Now if light were radiated in the 'right' direction from $t4$, it would be absorbed by and reflected from polygon C. Were light radiated in the left direction from $t5$, it would (presumably) pass through object C and reach object B - 'presumably' because this corresponds to no real physical situation, and is a heuristic.

3.3 Illuminating the VLF

All objects have been rendered into the VLF, but at this stage there is still no light - the light field is virtual. Light is introduced into the environment only by emitters. An emitter may be 'switched on', thus illuminating the environment. When an emitter is switched on, all the T-intersections for rays that intersect it have their radiance values set according to the properties of the light source. In Figure 3, object B is an emitter. The interval (T_4, T_5) is discovered, and in particular T_5 becomes:

$$T_5 = \{t_5, B, \text{left}, r\}$$

where r represents the radiance belonging to that interval.

However, it is known that this ray, in this particular direction strikes object A at position given by parameter t_4 along the line. Object A has certain material properties that determine its BRDF, which in turn determines rays that must be reflected from A at this point, and the radiance along those rays. Each of these rays may now be treated identically to the original emitted ray - their direction is known, and hence the (u, v, s, t) values can be found, the ray picked out of \mathcal{R} , the appropriate interval found, and so on.

Imagine this method were being used to achieve the same result as ray tracing, and that all objects were perfectly specular reflectors. The interval (T_4, T_5) strikes object A at parametric value t_4 on the line. The direction of the specularly reflected ray at this intersection point can be found, and will correspond to ray R . This has a T-intersection sequence, and the particular interval in that sequence containing the point at t_4 can be found. That interval may terminate at another object, and again the specularly reflected direction can be found, the corresponding R and T-intersection sequence derived, and so on. Of course, each newly found interval has its radiance entry adjusted to take account of the irradiance along the ray that spawned it.

Imagine this method were being used to implement radiosity, and that all objects were perfectly diffuse reflectors. Once again an interval terminates at an object in a particular direction. For a perfectly diffuse reflector all rays in the hemisphere over the hit point are affected. Each of these can have their radiance updated, and the paths followed.

Realistic scenes contain both diffuse and specular reflectors. The method introduced here of course deals with both simultaneously - there is no special issue in coping with the combinations of light transport. The material properties of an object determine which other ray intervals should be followed whenever an interval with associated radiance terminates at an object.

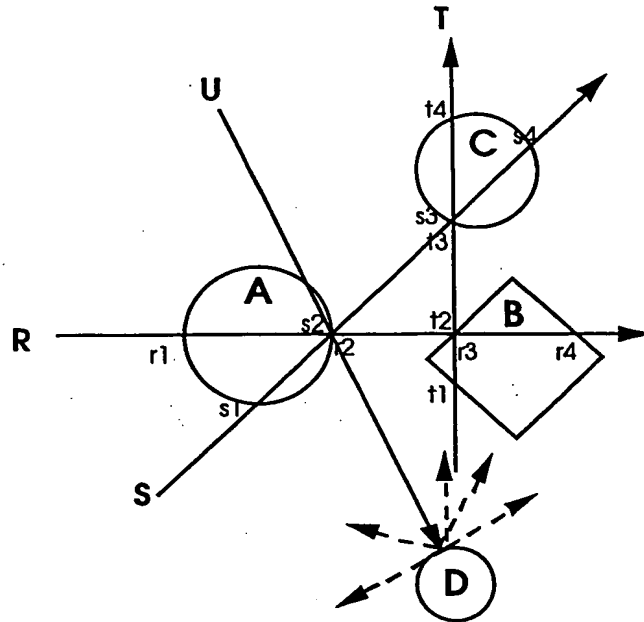


Figure 5
Specular and Diffuse Reflection

Consider Figure 5 with object A as a light emitter. The Figure shows four rays belonging to \mathcal{R} - ray R, S, T, and U. Each object has been rendered into the light field, and the rays partitioned accordingly into T-intersection sequences. Now light source A is switched on. The interval $[0, r_1]$ and $[r_2, r_3]$ become lit according to the properties of this emitter. The interval $[r_2, r_3]$ terminates at object B. Suppose B is a perfectly specular reflector, ray T corresponds to the direction of specular reflection. Along ray T, interval $[t_2, t_3]$ is the appropriate one, and its radiance is modified according to the reflective characteristics of B and the energy along $[r_2, r_3]$. Ray interval $[r_2, r_3]$ terminates at object C also a specular reflector. Ray S is the one corresponding to the direction of specular reflection (in fact the 'left' direction of ray S). Interval $[s_2, s_3]$ terminates at object A, and ray U corresponds to the direction of specular reflection. The relevant interval terminates at object D, which is a diffuse reflector, and all rays in the hemisphere above the intersection point are then treated in the same way.

Notice that there is *no searching for rays in this method*. If a full and continuous representation of \mathcal{R} and \mathcal{S} had been constructed by rendering all objects into \mathcal{S} , then any required reflection path determined as a result of a ray striking an object is already available, and the particular segment of that ray containing the intersection point is known. Hence the next object to be hit along the new path is immediately available and the intersection point is known.

The closest to this is discrete ray tracing, but even in that method a ray has to be traversed through the voxel space until it hits a voxel that belongs to an object. In discrete ray tracing, all rays start from the image plane. Here though there is so far no observer; the light field itself is being constructed in a view independent manner. However, once the

light field has been constructed, from the discussion in earlier sections it is clear that a virtual eye can be placed into the scene in order to produce a 2D image for any virtual eye position and orientation.

Of course, in practice, there is no complete and continuous representation of \mathcal{R} - instead a discretised one. There is no way that 'all rays in the hemisphere over a point' can be followed - there are uncountably infinite such rays. An exact ray required for a perfect specular reflection will exist with probability 0 in the discrete representation. However, we leave this discussion to Section 4. In this section we have shown how in principle a complete representation of the light field can be constructed, and one which accounts for all space within the environment, even inside objects. There is no conceptual restriction to 'free space' as is the case for the lumigraph approach of Section 2.7.

4. Practical Considerations

In Section 3 the fundamental ideas of creating a VLF were considered in abstract terms. In this section practical issues of representation and implementation are discussed. From a philosophical point of view the most important aspect of the approach suggested in this paper is that the VLF is the fundamental entity of interest in computer graphics, and its representation is the core data structure of computer graphics. In this way of thinking about computer graphics, the VLF replaces the 2D frame buffer as the core entity. The frame buffer is needed 'only' to allow real humans to 'see through the eyes' of a virtual creature inside the VLF - and just a feature of today's display technology - since some method is needed to produce 2D images corresponding to the images formed on the 'retina' of the virtual observer's eyes. Future graphics hardware may have the VLF directly encapsulated (no doubt in compressed form), and the vast bulk of the computation will be to change the individual entities (the T-intersections) that form the VLF. This different way of thinking about computer graphics is vital to keep in mind when reading the following sections.

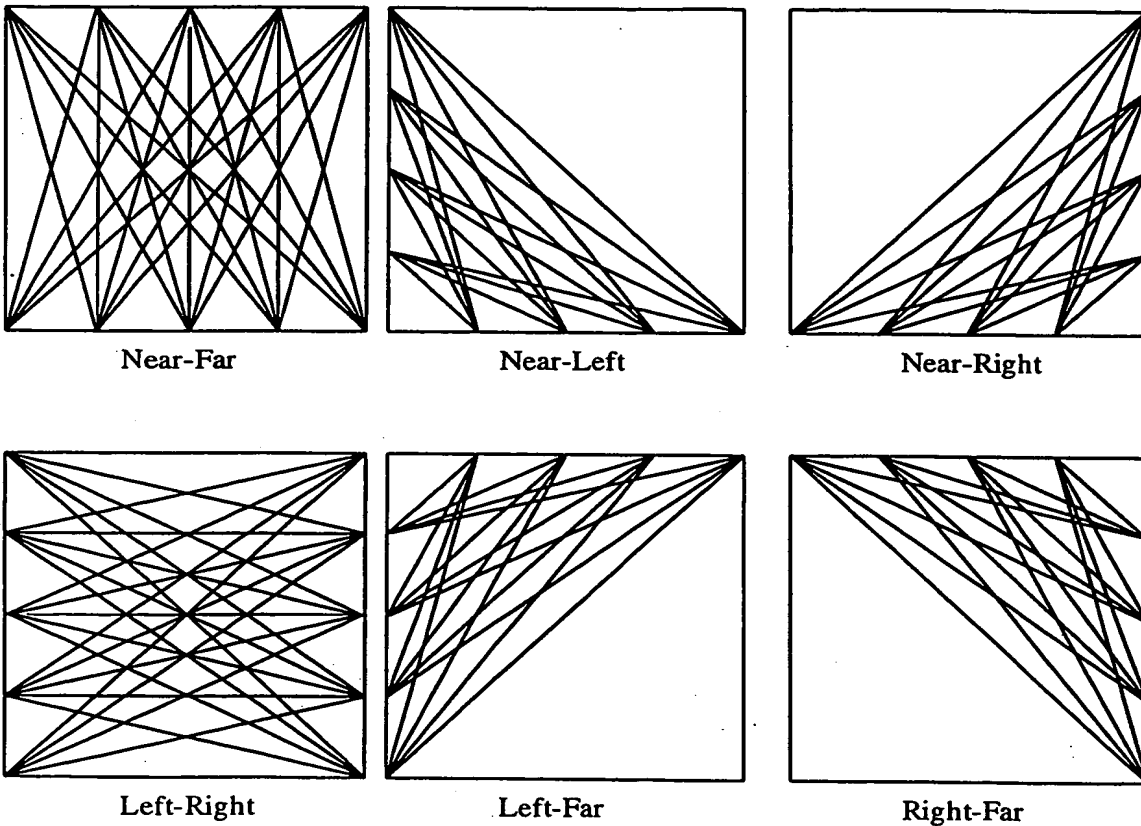
4.1 Representation of the VLF

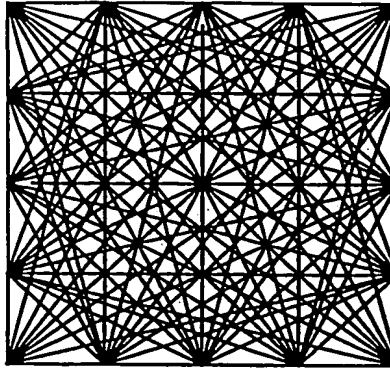
The VLF must be discretised for computational purposes. What is required is a parameterisation for \mathcal{R} , representing a collection of rays of the form $R_n(u,v,s,t,d)$, where u,v,s,t and d are as defined above. The index n refers to the overall size of the representation. Let \mathcal{R}_n refer to the set of all rays in this discrete representation. This must have the property that is for any $R \in \mathcal{R}$, and any $\epsilon > 0$, it is possible to find a ray $S \in \mathcal{R}_n$ such that $|R-S| < \epsilon$ for large enough n . A second property for \mathcal{R}_n is that for any $R \in \mathcal{R}$ and fixed n , there must be a (small) constant time look-up operation to find the ray $S \in \mathcal{R}_n$ that is 'closest' to R on some metric. The first requirement is simply that $\mathcal{R}_n \rightarrow \mathcal{R}$ as $n \rightarrow \infty$. The second requirement is that there must be no searching involved in finding the closest ray in \mathcal{R}_n to any given ray in \mathcal{R} . Using these criteria it should be possible to compare the efficiency of different representations, since there are obviously many possible choices for discrete representation. However, this is not discussed here.

Levoy and Hanrahan (1996) and Gortler et. al. used the two-plane parameterisation. It is not possible to represent all directions just with one set of parallel planes - at least three

(orthogonal) sets are needed, corresponding, for example, to planes $x=0$ (left), $x=1$ (right), $y=0$ (near), $y=1$ (far), $z=0$ (bottom), $z=1$ (top). Their argument in favour of this parameterisation was computational efficiency, and also that it results in a reasonably uniform coverage of ray space. It clearly satisfies the criteria above - n represents the number of grid points on the various planes, and assuming that the grid-points are evenly spaced the second criteria is also met.

In the first implementation of the VLF method a modified representation was used. This is illustrated for 2D in Figure 6, where each edge of a surrounding cube is divided into n equally spaced grid points. Each grid-point is connected to every other grid-point on the remaining edges.





All

Figure 6

Discrete VLF representation (4 subdivisions)

In 3D each face of a surrounding cube is subdivided into an n^2 set of grid points. Each is connected by a ray to each grid point on all of the other faces. There are 15 arrangements of such face-to-face projections. Let N, L, R, B, T, F stand for the near, left, right, bottom, top, and far planes. Then the 15 face-to-face projections are NL, NR, NB, NT, NF, LR, ..., BT, TF. In fact there are only two 'canonical' projections, NL and NF - all of the others are rotations and/or reflections of these first two. Similarly, in the two-plane parameterisation there is only one canonical projection. These 15 face-to-face projections also define the 'right' direction along each ray - for example, a NL ray has its 'right' direction from the Near ($t=0$) to Left ($t=1$) plane.

The reason for the choice of this particular discretisation was to cut down on the number of rays needed to obtain a wide coverage of the ray space compared with the two-plane parameterisation. No intrinsic benefits of this parameterisation are claimed, other than computational convenience.

Figure 6 illustrates the coverage of ray space for the 2D version of this representation. Clearly it is better more towards the centre of the scene compared with the periphery. Figure 7 plots for a 20 point subdivision the angle of the ray with the u axis for each u point. This compares favourably with similar plots produced by Levoy and Hanrahan for the two-plane parameterisation.

In the 3D case, if the number of grid points is n^2 , then there are $15n^4$ rays in this form of \mathcal{R}_n . For example, for $n = 16$, the number of rays is 0.9375MB, and for $n=32$ the number is 15MB.

Any R in \mathcal{R} can be trivially clipped against the bounding scene box to determine which one of the 15 face-to-face projections it belongs to, and the corresponding (u,v) and (s,t) intersection points. The ray in \mathcal{R}_n is chosen by the nearest grid point on each face.

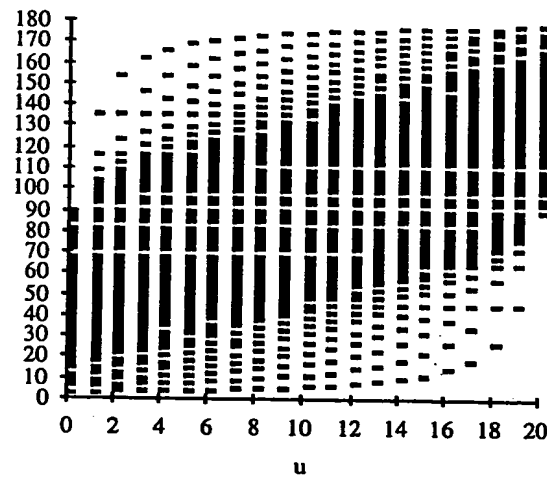


Figure 7
Ray Space Coverage in 2D (n=20)

4.2 Rendering Objects into the VLF Using the Two-Plane Approach

An important feature of the method is that there is no rendering of objects in the normal sense in computer graphics. Instead objects are rendered into the VLF only, rather than into the 2D frame buffer. This section briefly describes how this may be accomplished efficiently. In Section 4.2.1 general approaches are shown that will work for the two-plane parameterisation or the modified version. In Section 4.2.2 a method is shown whereby polygons and polyhedra can be rendered into the VLF using 2D line and polygon fill algorithms (of the type usually embedded in graphics hardware).

4.2.1 General Approaches

Two related problems are considered. Given any canonical face-to-face projection (eg, NL, or NF) find the set of all rays 'through a point' and also the set of all rays that intersect an axis aligned box. In fact no ray at all might exactly pass through any given scene point, so the problem is better stated as the set of all rays that approximately pass through a given point.

The lumigraph approaches also tackle this problem, but by finding all image projections, and only recording the radiance values. Here, recall, the purpose is to subdivide the rays into intervals to form the T-intersections. The lumigraph approach only considers such projections in 'free space' - here there is no such restriction. For any arbitrary object, find its smallest axis aligned bounding box. Each ray that intersects the box is further intersected with the object, so that the rays are ultimately partitioned by the object itself.

A theoretical treatment of the characteristics of the set of all rays through a point, a line and a triangle can be found in (Gu, et. al., 1997). For example in the 4D two-plane

parameterisation space, the set of all rays through a point in 3D space is a 2-dimensional affine subset.

(a) The Set of All Rays Through a Point

Here we consider only the two-plane parameterisation in its canonical NF form. There are similar results for NL, although slightly more complex. Suppose the distance from the near to far plane is m . Then coordinates on the near plane are of the form $(x_N, 0, z_N)$ and on the far plane (x_F, m, y_F) .

Suppose the point of interest is (a, b, c) . Then any line starting from the near plane and passing through this point has parametric form $(x_N + t(a - x_N), tb, z_N + t(c - z_N))$. This intersects the far plane where $t = m/b$, and therefore the far plane intersection point is:

$$\left(x_N + \left(\frac{m}{b} \right) (a - x_N), m, z_N + \left(\frac{m}{b} \right) (c - z_N) \right) \quad (8)$$

In terms of the coordinates on the far plane:

$$x_F = A + Bx_N$$

$$z_F = C + Bz_N$$

where

$$A = \frac{ma}{b}, C = \frac{mc}{b}, B = \frac{b - m}{b} \quad (9)$$

In practice the possible range of values for x and z will be bounded. By considering any line from (m, m, z_F) through (a, b, c) , and any line from $(0, m, z_F)$ through (a, b, c) it is easy to show that

$$\max \left[0, \left(\frac{a - b}{m - b} \right) m \right] \leq x_N \leq \min \left[m, \left(\frac{m}{m - b} \right) a \right] \quad (10)$$

with a similar result for z_N .

In the two-plane parameterisation x and z may be treated independently for this problem, so that (9) and (10) allow a Bresenham style algorithm to compute all the required rays. For the NL representation, this is not quite so straightforward since x and y are not independent.

(b) The Set of Rays Through an Axis Aligned Box

Finding the set of rays through an axis aligned box can be decomposed into two parts, finding near and far ranges of values for x , and for each determined ray, finding the near and far ranges for z .

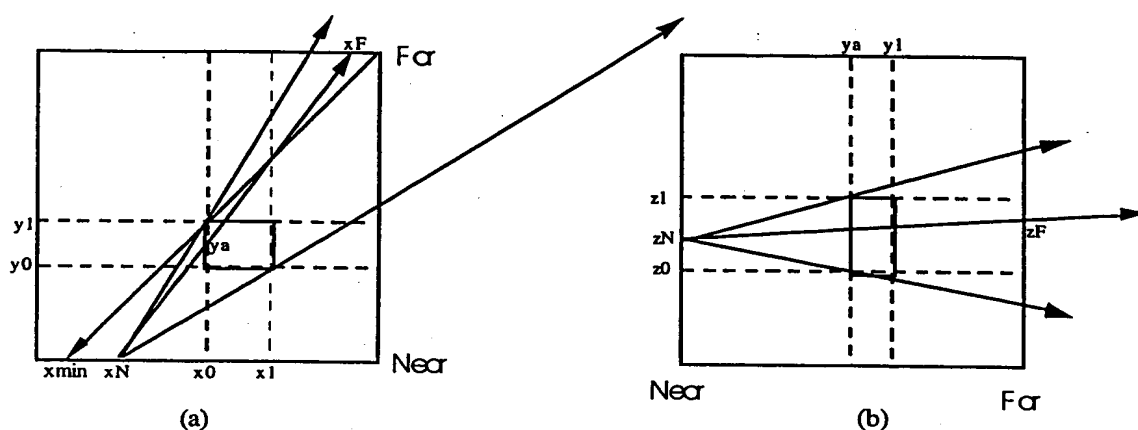


Figure 8
All Rays Through a Box

The method is illustrated in Figure 8. The box is defined by $x_0 \leq x \leq x_1$, $y_0 \leq y \leq y_1$, and $z_0 \leq z \leq z_1$. Consider x_N in the range $[0, x_0]$. By projecting from the point $(m, m, _)$ through $(x_0, y_1, _)$ (where the underscore represents a 'don't care' value) it is possible to find the lower bound for x_N (which may be 0 of course). For any x_N in this range, projection through the edges of the box corresponding to $(x_0, y_1, _)$ and $(x_1, y_0, _)$ gives the valid range of x_F values on the far plane, and for any x_N , x_F can easily be determined (Figure 8(a)). Each x_N , x_F pair represents a plane that slices through the box, forming a rectangle. This is shown in Figure 8(b), and may be used to determine the z ranges, in a similar manner. Suppose as shown that the range of values for z being considered is $[z_0, z_1]$. Then the valid range of z_F values can be determined by projecting through $(_, y_0, z_0)$ and $(_, y_0, z_1)$. The coordinate y_0 was determined by intersection, shown in Figure 8(a). For any z_N value, the corresponding z_F can easily be found.

This method therefore treats each x interval in turn, less than x_0 , between x_0 and x_1 , and greater than x_1 . For each such interval and generated x_N , x_F pair, it successively treats each interval for z_N . In this way all rays intersecting the box can be found. Although this has been described as if all the variates are continuous, it should be recalled that of course the grid points would be found, and it is these that correspond to the rays.

This method actually finds all projections on the far plane through each valid grid point on the near plane - hence its similarity to the projections used in the lumigraph approach should be clear. However, in this case it is achieved by explicit enumeration of the coordinate values, rather than by perspective projections. The other difference is that having found a ray through the box, this ray is then intersected with the object that the box encloses, and the T-intersections are formed if the ray does in fact intersect the object.

A concern might be that this is an extremely slow process, being equivalent to a very large number of 2D projections and rasterizations. However, in fact the 'rendering' of an object into the VLF is, in practice, likely to take the same order of magnitude of time as conventionally rendering an object into a frame buffer.

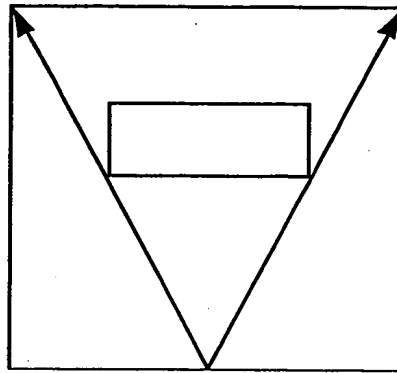


Figure 9
A Rendered Object Covering the Display

Figure 9 shows a situation where a viewpoint is on the near plane, and an object projected to cover a display the size of which is supposed as the same as the side of the enclosing box. If this were conventional rendering, the number of pixels rendered would be, say, N^2 , if N were the number of pixels in the rows and columns. Now suppose in the same situation, the same object were rendered into the VLF (considering the NF projection only). If there were M grid points along the rows and columns then the number of rays found would be M^4 . However, in practice M is going to be significantly smaller than N . The processing time, in each case including only finding the pixels or rays (but not operating on them), will be proportional to N^2 and M^4 respectively. If $M \propto \sqrt{N}$ then the times will be of the same order of magnitude.

This is not an unreasonable assumption. It suggests that the time to project an object into a pixel resolution of 1024^2 would be roughly the same as the time to render an object into a VLF of size 32^3 . When a pixel is enumerated there is further work involved in visibility and shading. When a ray is enumerated it then must be further intersected by the object itself, and the interval tree partitioned if there are intersections. It is difficult to compare the two processes because of the considerable hardware support for the pixel based rendering.

4.2.2 Rendering Polyhedra into the VLF using the 2D Polygon Fill Algorithm

This section shows how to render arbitrary regular closed polyhedra into the virtual light field. The polyhedra should satisfy Euler's equation that $V-E+F=2$, where V is the number of vertices, E is the number of edges, and F is the number of faces.

In the two-plane parameterisation the rays that are used to construct the VLF may be considered as sets of parallel lines. For example, the Figure 10 shows a 2D analogue of part of the NF construction. The overall NF construction is a superposition of all such sets of parallel lines.

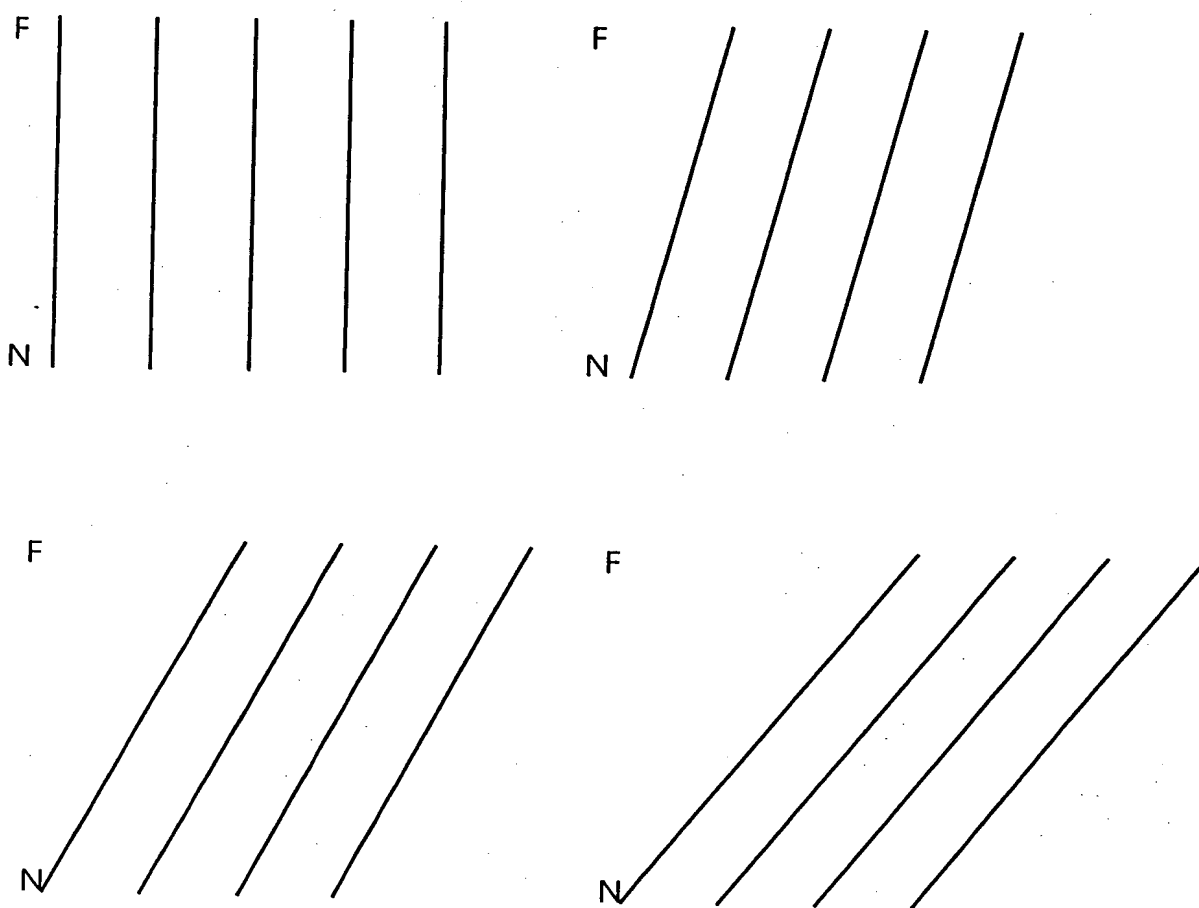


Figure 10
(Partial) Decomposition of NF into Sets of Parallel Rays

Clearly the VLF as a whole is a superposition of sets of parallel rays in 3D. Call any particular set of parallel lines a **parallel subfield** of the VLF.

In order to form the VLF the bounding box sides have each had a rectangular grid imposed, and each grid point has been connected to each other grid point around the box (where feasible). This is called an N^3 grid arrangement, where each box face has been subdivided into an $N \times N$ grid.

Consider any parallel subfield, for simplicity of explanation, a case when all the rays are axis aligned with respect to the scene bounding box, for example in the LR direction. At any grid height, the set of parallel rays will form a set of horizontal parallel lines.

Suppose there is a polyhedra in the scene, then at any height between the lower and upper bound of the polyhedra a polygon will be formed, intersected by the set of parallel lines. This is shown in Figure 11.

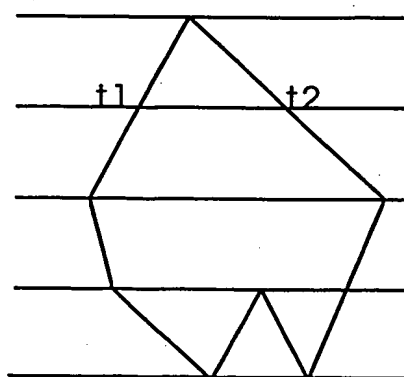


Figure 11

A Cross Section of a Polyhedra Intersected by the Rays in the Parallel Subfield

Note that the polygon vertex heights are appropriately rounded so that they fall on the parallel lines.

The standard polygon fill 'scan line' algorithm will incrementally find all the intersection points across the horizontal lines, such as t1 and t2 shown in Figure 11. This algorithm forms the basis of modern 3D graphics hardware. It is explained, for example, in Foley et. al. (1990).

The above only considers a particular slice of the parallel subfield of the VLF. However, it is simple to construct an algorithm that will process the entire subfield. This is a 3D adaptation of the same 2D polygon fill algorithm.

Each grid height defines a plane, and there will be N such planes for an N^3 arrangement. Let (x_1, y_1, z_1) to (x_2, y_2, z_2) represent a polyhedra edge with $z_2 > z_1$. Then this edge is represented as:

$$e = (z_1, x_1, y_1, dx/dz, dy/dz) \quad (11)$$

called an e-tuple, where $dx = x_2 - x_1$, and similarly for dy and dz .

There will be an edge table represented by a one-dimensional array $ET[]$ indexed by the z -coordinates, and each element will consist of a sequence of such tuples. The tuple shown in (11) would be appended to the sequence $ET[z_i]$.

Suppose that at height z , the intersection of an edge with the plane is (x_i, y_i, z) . Then at height $z+1$ the intersection will be $(x_i + dx/dz, y_i + dy/dz, z+1)$ - thus allowing an incremental computation of intersections along any edge.

The algorithm then proceeds as follows:

1. Adjust all z -vertex values of the polyhedra so that they fall on parallel line boundaries.
2. Find the lower bound of the polyhedra. It identifies a set of edges currently intersecting the corresponding plane. This will either form a single point, or a set of points. In the latter case the 2D scan line algorithm can be executed over this set of points as a polygon to find the relevant intersections.
3. The set of edges intersecting the current z -plane is called the active edge set (AET).
4. Move up to the next z -plane. Delete any edges in the AET which have their z -entry equal to the current z height.
5. Update the x and y intersection values to correspond to this z -height by using the dx/dz and dy/dz increments.
6. Bring into the AET from the ET any edges which have their lower bound starting at this z -height.
7. Perform the 2D scan-line algorithm on the polygon formed by the intersections with the current plane.
8. Repeat all the above until the upper bound of the polyhedra is reached.

It is important to realise that 'perform the 2D scan-line algorithm' does not mean that all 'pixels' between intersection points are shaded as would be the case in a traditional application of this algorithm, only that the algorithm is used to find the intersection points.

The argument above has assumed axis aligned parallel rays. However, the same algorithm can trivially be applied to any subfield of the VLF by applying an affine transformation to make the corresponding rays 'horizontal'. Of course, in practice a transformation would be applied to the vertices of the polyhedra rather than to the rays themselves.

The above shows how it is quite simple to adapt 2D scan-filling (and matrix transformation) hardware to render a polyhedra into the VLF. When the object is not a polyhedra, then the same method can be used to render any bounding polyhedra, such as

the bounding box of the object, and each ray discovered that intersects the bounding polyhedra is then further tested against the actual object itself.

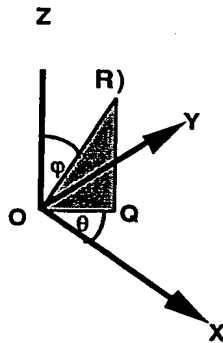
Sometimes it may be necessary to consider a single polygon rather than a polyhedra. It should be immediately clear that a single polyhedra will be a line segment (in the convex case) and a set of line segments (in the non-convex case) for any parallel subfield at any grid height. (If the polygon is parallel to the grid height then the intersections would not be relevant - since this would be equivalent to light striking the polygon edge-on). Since the polygon reduces to line segments, again efficient (hardware compatible) algorithms exist for finding the set of intersection points, such as the famous Bresenham algorithm.

4.3 An Alternative Representation

A representation that is preferred to the two-plane parameterisation has been developed, which has better properties for rendering objects into the VLF. This representation was been introduced by other authors in a different application, in the context of computing approximate visibility between portions of a 3D scene (Chrysanthou et al., 1998).

4.3.1 The Parallel Subfield Representation

Any ray direction can be represented in spherical coordinate form, as two angles (θ, ϕ) .



Consider, for example, the special case of rays parallel to the z-axis (vertical rays), with $\phi=0$ (in this case θ can be anything). Take a cuboid volume defined by $-1 \leq x \leq 1$, $-1 \leq y \leq 1$ and $-1 \leq z \leq 1$. Impose a rectangular grid on the xy plane, with n equally spaced subdivisions, the width of each interval being $2/n$ along each of the x and y axes. Discrete coordinates are defined by the centres of these intervals. This imposes a discrete coordinate grid on $-1 \leq x \leq 1$, $-1 \leq y \leq 1$, referenced by (i,j) . The real coordinate on the xy plane corresponding to (i,j) is therefore:

$$(x_i, y_j) = \left(\frac{2i+1}{n} - 1, \frac{2j+1}{n} - 1 \right), i = 0, 1, \dots, n-1; j = 0, 1, \dots, n-1.$$

Use this to define the vertical rays $(x_i, y_j, -1)$ to $(x_i, y_j, 1)$. Hence this defines an $n \times n$ array of rays spanning the cuboid in a vertical direction.

Now let (θ, ϕ) vary to cover all directions in the hemisphere above the xy plane. (This also, of course, covers all directions in a sphere around the origin). In particular, to cover this hemisphere, the following ranges can be used.

When $\phi = 0$, θ is not relevant.

For $0 < \phi < \pi/2$, $0 \leq \theta < 2\pi$.

For $\phi = \pi/2$, $0 \leq \theta < \pi$.

Each (θ, ϕ) combination defines an $n \times n$ array of rays, in the given direction, and may be thought of as a rotation of the vertical case.

Now (θ, ϕ) can be discretised as integers u and v . Let m be the number of subdivisions of 2π radians, and define $\alpha = 2\pi/m$, where m is an integer multiple of 4. Then for any u and v :

$$\theta = u \alpha, u = 0, 1, \dots, m-1$$

$$\phi = v \alpha, v = 0, 1, \dots, m/4.$$

The ranges of values of u and v are therefore:

When $v = 0$, u is not relevant.

For $v = 1, 2, \dots, (m/4)-1$, $u = 0, 1, \dots, m-1$

For $v = m/4$, $u = 0, 1, \dots, (m/2)-1$.

4.3.2 Computing the Representation

In order to satisfy the criteria for being of use as a VLF representation, it is necessary to show that any ray direction can be approximated as closely as required for a large enough representation, and that any particular ray represented in the discretisation can be found in constant time. The latter case is straightforward, since the rays can be stored, for example, as a 4D array, consisting of $n^2 m^2 / 4$ entries with lookup $[i, j, u, v]$.

In the following discussion, it may be assumed without loss of generality that the scene has been transformed to the unit cube defined by coordinates $(-0.5, -0.5, -0.5)$ to $(0.5, 0.5, 0.5)$. Note that this is conservatively chosen so that any rotation of this cube around any axis through the origin will always lie inside the cube defining the rays.

Let $R_x(\theta)$ and $R_y(\phi)$ be rotation matrices, which rotate in the positive direction an angle θ about the z-axis and an angle ϕ about the y-axis respectively.

Consider any ray with start point (x_i, y_i, z_i) and direction (dx, dy, dz) . The direction can be used to find (θ, ϕ) . The rotation matrix $R_x(-\theta)R_y(-\phi)$ will rotate the ray so that it becomes vertical (parallel with the z-axis). The (i, j) coordinates can now easily be found from this transformed ray, by using $(x_i, y_i, z_i) R_x(-\theta)R_y(-\phi)$.

For m large enough the true ray direction can be approximated to any degree of accuracy required. Similarly for n large enough the offset (i.e., the particular ray in the parallel subfield) can be approximated to any required tolerance. Hence this representation can be used to represent the VLF.

For completeness, it is worth showing how to obtain the real ray corresponding to a given (i,j,u,v) . From u and v find (θ,ϕ) . Hence $(x_i, y_i, -1)R_y(\phi)R_z(\theta)$ and $(x_i, y_i, 1)R_y(\phi)R_z(\theta)$ give the end-points of the ray in real coordinates.

4.3.3 Rendering into the VLF Using the Parallel Subfield Representation

This parallel subfield representation is ideal for rendering objects into the VLF (see Section 4.2.2). For each (u,v) combination there is a parallel subfield of rays. The corresponding (θ,ϕ) may be used to transform the object such that it is in a parallel subfield with vertical rays (in particular, transforming the object by $R_z(-\theta)R_y(-\phi)$ will accomplish this). Now the method given in 4.2.2 can be used to very efficiently find the set of (i,j) values corresponding to intersections of the object with these vertical rays. The rotation does not change the relationship between the rays and the object, so that all the results found for the vertical parallel subfield pertain to the original subfield.

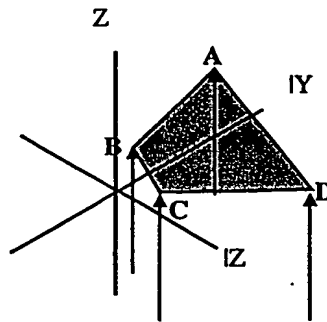
For example, in the case of a polygon, each j value within the range of the polygon, corresponds to a plane that will intersect the polygon in a straight line. The i -values and t -intersections along the ray can be found using a small adaptation of a traditional incremental 2D line rasterisation algorithm.

In the case of a polyhedra, each j will correspond to a plane that will intersect the polyhedra in a polygonal slice, and again an incremental polygon fill algorithm may be used to identify the i -values and intersections.

4.3.4 A More Efficient Method

There is a still simpler method for rendering polyhedra and polygons into the VLF, that is not simply similar to, but exactly the same as current hardware for rendering polygons into 2D framebuffer.

Consider any one parallel subfield, for convenience the vertical one (i.e., where all rays are parallel to the z -axis).



The figure shows an example of one polygon, and four of the total set of vertical rays. Clearly, each vertex of the polygon is specified as a coordinate (x,y,z) .

Each ray is specified by an (x,y) position (think of these as equivalent to 'pixels' on the XY plane), and a 'depth' (z). Hence the standard polygon fill algorithm can be used, where the (x,y) positions corresponding to the inside and boundary of the polygon projected to the XY plane gives the rays, and the z -values give the intersection points. The intersections along the rays in terms of the t parameter is simply derived from z . For example, if each ray is of the form $(x,y,-1)$ to $(x,y,+1)$ then $t = (z+1)/2$. The z -values can be found by incremental interpolation in exactly the same manner as would be calculated for a 'z-buffer' in the standard polygon fill algorithm.

For a polyhedra, again the standard 2D algorithm can be used with the edge table consisting of all the edges of the polyhedra.

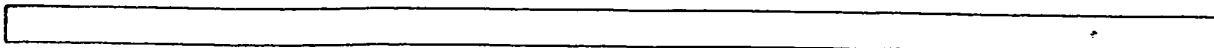
The above describes what happens for the parallel subfield consisting of the vertical rays. However, given any parallel subfield, the polygon or polyhedra can be rotated so that the rays become vertical, and then the vertical algorithm used as above.

4.4 Lighting

Once all objects are rendered into the VLF, lights may be 'switched on', and VLF rays then carry radiance. There are two approaches to achieving this - a depth first recursive approach, and a breadth first progressive refinement approach. We consider each in turn.

4.4.1 Recursive Method

This is carried out in a manner similar to traditional ray tracing, and is illustrated in the two functions `lightObject`, and `emitLight`.



```

lightObject(Object O){
  for each ray R intersecting object O{
    T = interval tree for R;
    /*suppose that each radiating direction is from T1 to T2*/
    for each interval (T1,T2) in T that includes O {
      set the radiance in T1 determined by O;
      emitLight(R, T1, T2);
    }
  }
}

```

The first of these enumerates each ray intersecting the object, and obtains the corresponding interval tree holding the T-intersections. For each interval that includes the object at one end or the other, it sets the radiance along that interval, and then calls the emitLight function on the ray and interval. For simplicity in this description the issue of the ray 'directions' is ignored, it being assumed that the object is always located in the left-most T-intersection of the interval. The lightObject function is called for each object that is a light source.

```

emitLight(Ray R, T-intersection T1, T-intersection T2) {
  Object O2 = object in T2; /*this is the object hit by the ray*/
  /*use R and BRDF(O2) to compute the set of rays to be reflected from O2*/
  for each reflected ray S caused by R hitting O2 {
    T = interval tree for S;
    for each interval (U1,U2) in T that includes O2{
      update the radiance in U1 determined by O2;
      emitLight(S,U1,U2);
    }
  }
}

```

The emitLight function takes as input a ray and the interval (T₁,T₂) along it that has just been lit prior to invoking this function. The ray is pointing towards the object O₂ in T₂ (if any). This object has material properties that will determine a set of rays that should be generated by this event, and the radiance that should be carried along these rays. For each such ray (S), the interval tree is obtained, and again for each interval that includes O₂, light is emitted along that interval. O₂ is now treated as a light emitter (of course, it is really a reflector).

There are several points to note about this method:

- There is no searching for rays. The closest to the true ray needed to represent a reflection direction is extracted from the discrete VLF in constant time.
- There should be a constant time lookup to retrieve the interval tree associated with any ray. For example, the interval trees could be stored in an array indexed by the face-to-face projection (NL, NR, ..., TF) and the (u,v,s,t) coordinates of the ray.

- The recursion can stop whenever the radiance to be set along an interval falls below a certain tolerance.
- Radiance is 'updated' in the sense that an interval may be visited several times (eg, for different types of reflection - specular or diffuse or the result of a transmission) and the new value is added to the existing value.

This scheme is very simple to implement, certainly hardly more difficult than a standard ray tracer. However, it has three disadvantages: first, it can reach enormous depths of recursion, and second, more importantly, it is not an easy scheme in which objects can later be added into or deleted from the scene (an issue to be taken up in Section 5). The third issue is that the entire recursion on all lit rays has to complete before anything is available for display.

4.4.2 Progressive Refinement

This method requires no recursion, but instead is a 'breadth first' approach, similar to progressive refinement radiosity (Cohen, et. al., 1988). The idea is to emit light from all sources first of all. The objects that receive light energy are simply noted, but no action is taken. In the next phase, each object that was marked as having received light is visited, unmarked, and all the received light is reflected. The reflection paths are not followed, but again, the objects that were the recipients of this phase are marked. This cycle continues until there are no objects marked as receiving light but not yet reflecting it. This light that has been received by an object but not yet reflected is called 'unshot radiance' following the terminology in the radiosity literature. Note that this closely follows the interpretation of equation (4).

```
lightObject(Object O){
    for each ray R intersecting O{
        for each interval (T1,T2) radiating out from O along R{
            set the radiance in T1;
            set the unshot radiance in T1;
            O2 = object at T2;
            mark O2 as having received unshot radiance;
        }
    }
}
```

The data structure for a T-intersection is modified to include the unshot radiance entry to become:

{t, identifier, direction, radiance, unshot radiance}.

The lightObject function iterates through each ray that intersects the object. As before the intervals containing the object, and in the outgoing direction are found, and the radiance is set according to the properties of the emitter. If such an interval terminates at

another object, then the unshot radiance of the outgoing T-intersection is set to the radiance, and the object is marked as having received this unshot radiance.

```

reflectObject(Object O){
    unmark this object;
    for each ray R intersecting O{
        for each interval (T1,T2) incoming to O at T2 with unshot energy at T1 {
            /*using R and the BRDF(O) to find the set of reflected rays*/
            for each reflected ray S{
                for each interval (U1,U2) radiating out from O along S{
                    update the radiance in U1;
                    set the unshot radiance in U1;
                    O2 = object at U2;
                    mark O2 has having received unshot radiance;
                }
            }
        }
        set the unshot radiance at T1 to 0.0;
    }
}

```

The reflectObject function is called for any object that has received light energy but not yet reflected it (the 'unshot' radiance). First the object is unmarked (it may of course mark itself again later if it is concave). For each ray that intersects the object all incoming intervals that terminate at this object are found. If these carry unshot radiance then the rays that are reflected as a result of this are found, and the relevant outgoing intervals on these have their radiance updated, and their unshot radiance set, as before. Objects that are the recipients of this unshot radiance (i.e., those at the other ends of the outgoing intervals) are marked. At the end of this process, the original interval that caused all the reflections has had all its unshot radiance dealt with, and this is set to zero.

The main program sets up the VLF data structure, renders all objects into it, and lights all the emitters. Then a straightforward method to follow is to keep cycling through all the objects, reflecting any unshot radiance, until there is no object marked as having received unshot radiance. The objects should be considered in the order of decreasing total unshot radiance - so that the objects contributing the greatest amount of energy are dealt with first. A maximum number of cycles can be set. Also a tolerance must be used whereby if the unshot radiance falls below a certain threshold then it will not be reflected. At the end of each cycle an image of the scene could be formed if desired.

```

main()

set up the VLF;
render all objects into the VLF;

```

```

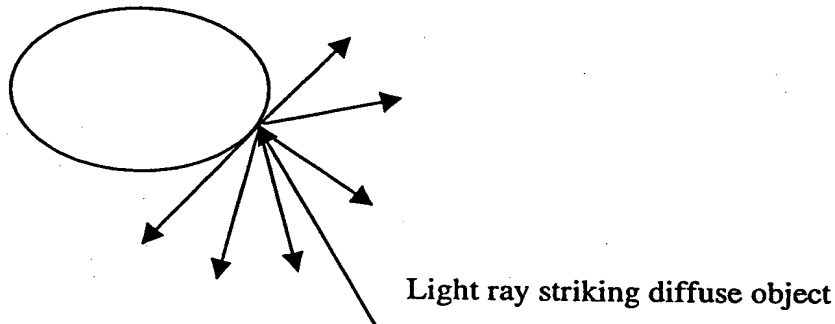
for each light emitter O lightObject(O);
for maximum number of cycles{
    finished = true;
    for each object O in decreasing order of total unshot radiance{
        if O has unshot radiance{
            finished = false;
            reflectObject(O);
        }
    }
    if(finished) stop;
}

```

4.4.3 Avoiding Computational Explosion for Diffuse Reflection

This section describes an aspect of the method that is extremely significant for practical application. It can reduce the processing time for computing the VLF when there are diffusely reflecting objects on an order of magnitude from weeks to hours or minutes for complex scenes.

When a light ray strikes a diffusely reflecting object, light energy is propagated along all rays in the hemisphere above the object at the intersection point.



This is obviously very computationally expensive, since there will typically be a large number of such ray-object intersections, and for each one all rays through the intersection point must be enumerated, the intersection interval found, and the energy put into the corresponding T-Intersection.

A much more efficient method requires only two passes through the object - i.e., all rays through the object are enumerated twice only. These two passes are called 'gathering' and 'shooting' (these names have been used in another context in the 'radiosity' method).

First suppose that each such object is tiled into small surface elements and on each element we store a representation of energy (initialised to zero). In fact it is important to

note that this approach of partitioning objects *is not used as such*, but it is a simple way to explain the method.

(a) Gathering

For each ray through the object carrying unshot energy into the object, we find the object surface element through which that ray interval passes, and accumulate the unshot energy into that element. Before accumulating it into the element this unshot energy is modified by the angle between this incident ray and the surface normal, as usual, according, for example, to Lambert's Law and the coefficients of diffuse reflection for this surface (or whatever other material property formula is used, for example, from a BRDF).

By the end of the 'gathering' process, all surface elements that received unshot energy contain an accumulated total energy that has entered that element.

(b) Shooting

For each ray through the object find the interval on the ray by which energy is to be carried out. Find the surface element through which this interval passes. Send out along this interval as its unshot energy, the energy (or whatever proportion of it is required) that is stored at this surface element.

Of course the usual resetting of unshot energy once it has been 'spent' is required, and the surface element energy representations must be reset to zero after all shooting is completed. However, this can easily be done without having to run through all the elements.

Now surface elements are not preferable as the actual method of implementation. The problem would be how many such elements to choose, their size, and also the complexity of algorithms to determine such surface elements (although it would be very similar to mappings used for textures).

A better method is the following, since it also corresponds to the resolution of the VLF. Each intersection point of a ray with an object of course corresponds to a unique T-Intersection along the normal to the object at that position. The accumulated energy can therefore be stored in that T-Intersection. This guarantees that the resolution used in the partitioning of the object is exactly the same as the resolution by which that object is 'known' to the VLF. Moreover it is a completely natural representation in the context of this overall method.

4.6 Viewing

In principle given a VLF viewing is simple. A virtual camera can be established anywhere in the scene. The camera will have a centre of projection, the convergence

point of all rays through the image plane. Equivalently, a pin-hole camera can be modelled. All rays entering through the pin-hole and striking the 'film' plane are found. The film plane is mapped to a 2D display window, and each ray that strikes the film plane affects the colour of a corresponding pixel.

The problem is that with a discrete VLF it is a probability zero event to find even one ray that will pass through the pin-hole, and even if there is one, again it may not hit the film plane within the required boundaries. The problem of generating enough rays in a discrete representation of light is the whole motivation for traditional 'backwards' ray tracing - tracing rays backwards from the COP, through the image plane and into the scene.

The solution to this problem actually provides a benefit - instead of a pin-hole camera a camera with a lens arrangement can be used. To distinguish this from a traditional camera model the term 'virtual eye' or just 'eye' will be used instead. This can be defined with any type of lens arrangement, whether the simplest, one 'thin' convex lens, or several 'thick' lenses. For simplicity a description will be given only of the method using a single thin convex lens. This is equivalent to having a box camera, with a finite lens instead of a pin hole, and the back-plane of the box being the film plane. Any ray that strikes the lens is bent, into the box, and a high proportion of such rays will also strike the film plane (image plane) at the back of the box.

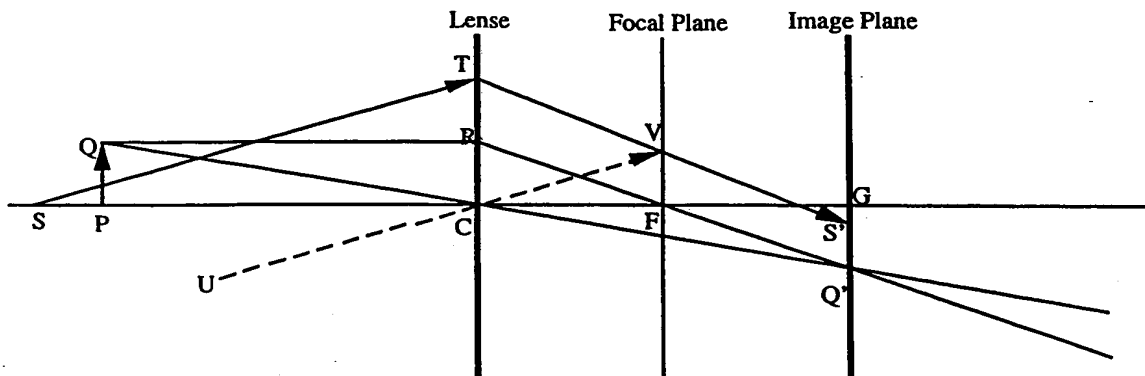


Figure 12
Configuration for a Box Camera with a Lens

Figure 12 shows a schematic for setting up such a virtual eye. The lens is a convex thin lens with centre at C, and focal plane through F. Suppose point Q in the scene is chosen as a point of interest. The ray QR is refracted through the lens to RF. A ray QC through the centre of the lens meets RF at Q' which is the image point for Q. This can be used to define an image plane. Ray ST is an example of a non-paraxial ray. It can be projected into the image plane by finding the parallel ray (UV) that passes through the lens origin, and then TV is the refracted ray. This strikes the image plane at S'.

Now the image plane can be mapped into a 2D window on the display. This window will contain a 2D array of pixels. For each ray that strikes the image plane the corresponding pixel is found, and its colour adjusted.

In practice, even for a window resolution of 100×100 it is extremely unlikely that rays will be generated that cover this resolution. Instead another scheme is used, again which has some advantages for anti-aliasing. The window is tiled into a rectangular grid - each rectangle then acts as a small photo-sensitive cell. (The tiling does not have to be regular, but, for example, the cells can be smaller near the centre and larger around the periphery). As rays strike the image plane, the corresponding cell is identified, and the radiance along the ray used to adjust the colour intensity of the cell. For example a simple average can be used, or a more sophisticated filter that distributes the intensity out over neighbouring cells in a Gaussian distribution.

The number of lit rays that strike the lens as the eye moves round the scene remains approximately the same. It is only this number that determines the time taken to project to a 2D image plane. This is the sense in which the method is described in the title as 'constant time'. The term 'constant time' can with justification be translated to mean 'in principle real-time' since there is a trivial operation for each ray, and finding the set of rays that pass through the lens is equivalent to 'rendering' the lens into the VLF. The crucial point is that once the VLF has been constructed, the time to render a globally illuminated scene is independent of the number of objects.

The virtual eye itself could be an object in the scene and thereby influence the overall distribution of light in the scene itself. However, a problem with this is that each time the eye moved it would affect the distribution of light, and therefore the VLF would have to be updated. The problem of dynamic changes is considered next.

5. Dynamic Changes

Up to now all the discussion has been in terms of walkthrough of a static scene. An argument has been given that this can, in principle, be accomplished in real time, since there is a small, constant time operation to map virtual light rays that intersect a lens into an image plane. However, interactive 3D graphics requires not simply walkthrough, dynamic changes of viewpoint, but also changes to the scene itself. There are two operations that must be possible - all others can be constructed from these: inserting a new object into the scene, and removing an object from the scene. In this section it is shown how these operations can be accomplished without reconstructing the entire VLF; however, they cannot be carried out in constant time.

Amongst all the methods discussed in Section 2, only progressive refinement radiosity admits the possibility of dynamic changes to scene geometry (again not in constant time). For each of the other methods, if a new object were added into the scene, the entire current solution would have to be abandoned, and the whole process repeated again. Of course special case circumstances could be defined where this would not be necessary, but here the goal is a general approach that would work for any circumstance. Chen (1990) showed how this is possible in using the progressive refinement algorithm in radiosity.

First consider adding a new reflecting object into the scene. The object is rendered into the VLF. In addition, during this process, each ray intersecting the object that carries

radiance has its unshot radiance entry set to correspond to its radiance entry. The object itself is marked, as described in Section 4.3. Also the interval where that same ray exits the object is found, and *negative* energy, i.e., energy to *subtract* is output along that ray interval, and any receiving object is marked as receiving unshot energy. (The fact that it is negative makes no difference to the subsequent computation).

For example, suppose in Figure 3 object B is inserted, and object A is emitting light. Then B would be marked as having unshot energy, and the energy received along the interval starting at t_4 in the 'right' direction would be stored as unshot. Since object B now blocks light along this ray, the interval in the 'right' direction starting at t_6 would have negative energy output along it, and any receiving object would be marked accordingly.

Then a new cycle is started, which can begin with this new object, since this is likely to be the one with the most significant unshot energy. The iteration continues, as described in 4.3, until no object is marked. Clearly, this is not a constant time operation, since the geometric and material properties of the object will determine the impact it has on the VLF as a whole.

Since an object can be added to the scene in this way, an object already in the VLF can be deleted. The first step is as before, the object is rendered into the VLF and the unshot radiance entries are set, but now as *negative* energies. Also, in the continuation of the ray that exits from the object, *positive* unshot energy is emitted (since now the ray is unblocked). This is essentially the opposite of what happens when an object is inserted. Again, a new cycle is started, but this time the object reflects negative radiance - its material properties are used to determine what energy to *subtract* from the outgoing reflected rays. The process continues again until no object is marked. At this stage, however, the object still exists in the VLF, though it reflects no light. The final step is to 'de' render the object from the VLF by removing the T-intersections that were generated by it from the interval trees associated with each ray that intersects it.

If an object were being transformed (for example, translated) then it would be wasteful to first go through the process of deleting it, and then the process of reinserting it - since there would be a huge number of rays that intersect both the object in its original position and in its new position. Hence the union of the old and newly positioned object can be found, but each ray treated appropriately to the circumstance when it is only intersecting the old object, intersecting partially the old and partially the new, and only intersecting the new object.

Similar techniques can be applied when the material properties of an object are changed, or when a light source is made brighter or darker. Additional positive or negative energy is cycled into the VLF until a new equilibrium distribution is established.

There is no claim here that these changes are potentially real-time operations - the significant point is that they can be done at all.

It is interesting to note that the standard computer graphics pipeline has never offered a 'principled' solution to the problem of dynamic changes, nor in fact to real-time

walkthrough. The solution obtained is one that relies on producing a system fast enough to re-render the entire scene, going through the whole pipeline for each object primitive (in fact triangles) at sufficient speed that the images can be regenerated at real-time frame rates. Of course, it does not matter whether or not the viewpoint or the geometry has changed between frames - if they have changed these changes will automatically be displayed.

6. Results

The method has been implemented on a Silicon Graphics O2, with 224MB main memory, but only using the *modified two-plane parameterisation, and without the method described in 4.4.3*. The greatest VLF resolution that has been possible is 32^3 (i.e., each scene bounding box face is divided into 32^2 grid points as the end-point of rays). Refer to 1KR as being 2^{10} rays and 1MR as being 2^{20} rays. Then a 32^3 resolution results in 15MR using the modified two-planes approach. This is too large, within these memory constraints, to generate the VLF for scenes with fully enclosing diffusely reflecting surfaces. Typically, many proof of concept examples have been generated using a 16^3 resolution, which results in just under 1MR in the VLF. Two examples are shown in the colour Plates². The display resolution used has been up to 100×100 super 'pixels' where each such 'pixel' is a small rectangular area. Only a simple averaging scheme has been used over the accumulated intensities in each super pixel.

The results are very encouraging. For these necessarily small scenes at low resolution, there are very clear global illumination effects, and each new image resulting from a change in view can be generated in a fraction of a second. Obviously the time depends on the number of rays that intersect the lens. There is no special importance to be attached to the actual time, however. The important point is that for any fixed combination of VLF resolution, virtual eye setup (in particular the size of the lens), and display resolution, the time to render an image will hardly vary from frame to frame.

The virtual eye is particularly interesting, in that it supports change in focus, and depth of field with no extra cost. This in itself can greatly add to the realism of the displayed images. As some observers have noted when viewing some of the images that have been generated, they look more 'alive' than the traditional sharp edged computer graphics renderings.

The images shown in the plates should be viewed from about one arm's length away. Plate A shows a room which has one light source and two spheres floating in mid air. The walls of the room are highly reflective as well as having a diffuse component. The VLF had resolution 16^3 . The number of cycles to convergence was 4 and this took 19 minutes and 59 seconds to complete. The 'pixel' resolution was 100^2 . Plate A1 gives an overview of the scene, clearly showing shadows and reflections. The distortions around the frame of A1 is due to the low sampling resolution of rays near the edges. In Plate A2 a depth of field effect can be seen - the light is out of focus. Plate A2 also shows a reflection of the green ball in the shadow of the blue one. Plate A3 is a view close to the light source. Plate A4 clearly shows reflections of reflections. The light source is reflected directly on

²The plates can be found at "<http://www.cs.ucl.ac.uk/staff/m.slater/Temp/Paper>" and *must* be viewed on a high resolution at least 24-bit display with maximum colour resolution.

the right hand wall. The light source is also reflected on the ceiling. This ceiling reflection is itself reflected in the right hand wall.

Plates B show four spheres: one light source and three specularly reflecting spheres, again with some diffuse reflection. The middle sphere is nearest to the view in Plate A1. The VLF resolution was 32^3 and the display resolution is 100^2 . The illumination phase completed in 5 cycles, which took 1 hour, 23 minutes and 59 seconds to complete. It could be argued that a ray tracing of the same small scene would take a much smaller time. However, ray tracing would produce one image; from the VLF any number of different views can be very quickly constructed. Plate B1 shows specular highlighting effects. The eye is more focussed on the two outer spheres than the inner one. In Plate B2 the image viewpoint has moved to the left, facing towards the purple sphere which is also the one most in focus. Notice how the green sphere and the light are out of focus. Plate B3 shows a view from the other side (although notice the gaps in rendering of the green sphere). The blue sphere shows a wide band reflection of the light source, a narrow band specular reflection of the light source, and a reflection of the green sphere. The Green sphere also has reflections of the blue sphere and the light source. Plate B4 shows an interesting lens effect. The light source is very close to the viewpoint, and lens distortions (recall that the convex thin-lens is only supposed to be used for paraxial rays!) has smeared the light source over the image (not unlike holding an object very close to your eyes and it blurs out over a wide field). In this image again we see reflections of reflections.

7. Parallel Architecture

There is clearly much scope for implementing the VLF method in a parallel architecture. There are three distinct phases. The rendering and reflection phases require coarse grain MIMD parallelism. Suppose there are n processors. Each processor is given a particular subset of the VLF to deal with. As a simple example, using the modified two-plane parameterisation, 15 processors might be employed, each dealing with one face-to-face projection (NL, NR, ..., TF). Alternatively subsets of the VLF can be chosen simply by the first processor dealing with particular subsets of rays across all faces. For example, if there were n processors, each one might deal with a different subset selecting each $(n-1)$ th grid point across the rows and down the columns. For the rendering phase each object is rendered by each processor in parallel.

In the reflection (or distribution) phase objects must be dealt with sequentially, starting from the light emitters. However, each object can be treated in parallel for each phase. There is an additional complication in this phase, since the 'best approximating' ray to a given true ray might belong to a processor other than the one that generated the request. Therefore there has to be a mechanism for transfer of requests for computation from one processor to another. This could be accomplished by a shared memory architecture, a message passing architecture, or preferably a combinatorial architecture where each processor is connected to each other one. A queue of requests would therefore form along each such communication channel.

The viewing phase can benefit from additional fine grain parallelism. Any ray discovered to intersect the lens is treated the same as but independently of any other ray, and

therefore a potential speedup can be obtained by groups of rays being treated in parallel. The results are accumulated into a frame buffer for filtering before final display.

8. Discussion

This paper has introduced a new method that can produce images of an interactive walkthrough of globally illuminated scenes at constant frame rates. Unusually for global illumination methods, it can also add and subtract objects without having to start the entire process over again.

Current graphics systems have the 2D frame buffer at their core. Since the frame buffer contains a pixelised representation of a 2D projection of the 3D scene, it is not surprising that most methods of rendering, are very much view dependent. As discussed in the previous section, this is true for current hardware accelerated 3D graphics systems - but hidden by the fact that for scenes up to a certain size, the images can be regenerated at about 20Hz or more. However, as the number of objects increases, so inevitably the real-time frame-rate is lost. More than this, typically the frame rate varies through time depending on the complexity of what is in view at any moment. As hardware performance increases, so the scene complexities offered to the hardware are increased - expectation rises. This motivates the next generation of hardware, that can cope with even more complex scenes. In the long run this is a pointless cycle - and will reach an absurdity where the number of polygons to be displayed outstrips the number of pixels³.

The current quest for image based rendering, while valuable in itself, is also an admission of this problem with conventional graphics systems. Image based rendering has the property that its rendering time is clearly independent of the number of objects: because objects are not rendered, only images. The most sophisticated version of this approach is the lumigraph method discussed above. However, this does not provide a solution to the real-time display of virtual globally illuminated scenes, because still some other method would have to be used to create the 2D images - such as radiosity, ray tracing, or a Monte Carlo approach. In the case of radiosity or photon tracing, these allow real-time walkthrough in any case; with ray tracing there is still the problem of the enormous amount of time it would take to render all of the required images. For synthetic scenes, there is a strong case that the method presented in this paper presents a more general and appropriate solution.

The system presented in this paper offers a different paradigm. The frame buffer becomes almost an irrelevance. Its place is taken by the VLF. It is important to understand the consequences of this. Once a particular instance of the VLF is chosen (i.e., a particular resolution) that is all there is. Readers will be tempted to think about 'interpolations' between rays. But this makes no more sense than to talk about 'interpolation of pixels' in the frame buffer: of course it is possible to interpolate between the values stored in two pixels, but where will the interpolated result itself be stored? - the given pixels are all that is available. So here, the particular discrete set of rays represented in the VLF is all there is: new rays being interpolations of other ones cannot be inserted - since the assumption

³ This was pointed out by Professor Fred Brooks Jr, of University of North Carolina, during a lecture to the undergraduate Computer Graphics course at UCL in December 1997.

is that the VLF has a hardwired encoding much as the array of pixels. Such a hardwired encoding is necessary so that the constant time access to any particular ray being the closest approximation to a true ray is maintained. The VLF should be thought of as a (set of) fixed 4D arrays of interval sequences (usually encoded as interval trees), representing subdivisions along a ray. Of course, methods equivalent to 'supersampling' in pixel space can be devised, or other methods for reducing ray-space aliasing, but in the last analysis only the rays available through the given VLF representation are available for use.

The VLF representation, because discrete, is an approximate one. Such approximation is inevitable - it has moved approximation to a higher dimension, instead of just the approximation of continuous entities on the discrete 2D pixel array. In fact the approximation seems to be quite acceptable across the range of scenes generated. Using a 32^3 VLF all correlations were found between the true rays generated during the lighting process, and the actual rays available in the VLF. The correlations were all extremely high, above 0.95. However, a systematic study is required to quantify the degree of error: not just the error in final image space, but also the error in distributed radiance.

As discussed above, the fundamental representation of a scene in the standard computer graphics approach is actually at the 2D level: stored as a view encoded in a 2D frame buffer. Volume rendering represents an advance on this, using 3D as the fundamental representation - visibility ceases to be the central issue, rather there is a direct simulation of the scene in the computer memory, as opposed to a geometric representation in terms of a scene data structure, and a separate 2D representation of a particular view of that scene. Although volume rendering has been argued to have many advantages over the conventional approach (Kaufman, 1996) it has never become mainstream. Somehow just stepping up from 2D to 3D doesn't provide sufficient computational advantage in comparison to the additional costs. This is because at a deeper level, the change is not significant - for example, with discrete ray tracing, there are certainly advantages compared with standard ray tracing - but the core algorithm remains unchanged, offering no promise of a significant breakthrough in rendering speed.

The reason for this is that just moving from 2D to 3D really does not make that much difference - because still it is objects that are represented and not the really important phenomenon in computer graphics, which is light. It is necessary to move from 2D to 5D (4D for the ray representations and one dimension along each ray) in order to be able to represent light. The cost involved in terms of memory is at least an order of magnitude greater than for the volume representation; but the benefits inherent in the new approach might far outstrip the costs. It remains to review these potential benefits of the new VLF approach to computer graphics:

(a) A method has been introduced that allows objects to be rendered into a virtual light field, without any constraints on viewing geometry, such as the constraint to convex regions as in the lumigraph approach. The method does not need to render 2D images in order to construct the VLF, rather the geometry of the objects is written into the VLF by partitioning rays according to the object intersection points.

(b) Light can be emitted from any type of object, and distributed through the VLF, based on the material properties of the objects involved.

- (c) Any type of object can be used, provided that it is possible to find an intersection between a line and the object, and to compute the normal at the intersection point. There is no restriction to polygons.
- (d) Radiance for outgoing rays from an object can be computed using any model, and a texture map associated with an object can obviously also be used to modulate the radiance calculation. This involves simply having a look-up from any point on the object into a separate colour-map or formula.
- (e) The result of conventional ray tracing can be simulated, simply by making each object a specular reflector (together with some light sources) in addition to reflecting light diffusely.
- (f) The result of conventional radiosity can be simulated, simply by making each object an ideal diffuse reflector (together with some light sources).
- (g) Any type of light transport can be simulated without special-casing - diffuse-specular, specular-diffuse, specular-diffuse-specular, and so on for any combination of any length. Each object simply reflects light according to its material properties, independently of what any receiving object might 'do' with that light.
- (h) 2D images can be extracted from the VLF in near constant time. The virtual eye can be represented with any kind of lens arrangement, thus allowing depth of field effects. The depth of field effects are without any additional rendering cost.
- (i) Walkthrough rendering time is independent of the number of objects that were used to create the scene. This offers a solution to the greatest challenge that there is to current day computer graphics, but without the need for image based rendering.
- (j) Objects can be added to or removed from the scene, with an incremental update to the VLF.
- (k) The method also supports volume rendering in the sense that objects are effectively represented as solids. A ray is partitioned by an object into segments outside and segments inside the object. Provided that colorimetric information were kept representing the interior of objects, any object can be 'sliced' and the exposed solid interior contribute to the VLF. There is nothing in the method that requires it to be limited only to surface representations.

9. Applications

There are many potential applications of the technology described in this paper.

- (1) The obvious one is for architectural walkthrough and the simulation of realistic lighting for existing buildings, or the lighting in a particular building (eg, for a stage play or opera).
- (2) An interesting one makes use of the possibility of any type of lens arrangement for the virtual eye. When a television production or movie is being designed it would be useful to be able to simulate in advance of constructing the film set what the scene would look like under various lay-out and lighting conditions and with different types of camera lens. The VLF approach provides a method for doing this.
- (3) Since there is little cost involved in producing an image from the VLF, the method is ideal for stereo viewing as might be used with a Head-mounted display for virtual reality. Currently, the entire viewing pipeline has to be executed twice for each view - one for the left eye and once for the right eye. Hardware systems best do this in parallel. The VLF method can produce multiple simultaneous views by having multiple virtual eyes.
- (4) A problem with virtual reality head-mounted display systems is that the point of focus is fixed. The new method, since it supports depth of field through manipulating the focal length, or the view reference point of the virtual eye, can adjust the focus according to the nearest object on the line of site of viewer. Focus can be continually and automatically adjusted as the viewer looks around a virtual scene.
- (5) The VLF is, of course, a digitally encoded representation of a scene through representation of its light. A digital encoding is ideal for transmission, with compression, for example, for broadcast television. A local set top box, hardwired to decode and extract images from the VLF could be used to display games or other entertainment to people in their homes. The same is true of course across the internet.
- (6) The radiosity method relies on breaking up objects into polygonal patches. The new method can simulate radiosity without any polygons at all.
- (7) Although the purpose of this paper is to present the VLF method as a new paradigm for computer graphics, there is also an application that could be used to improve the current graphics pipeline. This is to do with the problem of visibility culling: given any scene and view, find as quickly as possible the set of polygons likely to obscure other polygons in the scene. If these can be quickly identified, then they can be rendered first of all, and in the usual z-buffer algorithm other polygons that were obscured will never be rendered at all (since all z-depths would be greater than those already set in the z-buffer). Now an approximate solution to this problem would be to render all objects into the VLF, but without the lighting phase. Before rendering each frame in 2D check a set of rays in the VLF that correspond approximately to the viewing direction and field of view. Each ray corresponds to an interval in direction (T_1, T_2), and the object at T_2 would be a likely scene obscurer. This method can be employed with a relatively small VLF - only a few rays are likely to be needed to find the large occluders, and it is only the large occluders that are really of interest.

10. Conclusions

The VLF method is offered as a new paradigm for computer graphics. At the moment images it can produce although interesting in their own right, compare poorly with what can be produced in the conventional approach. What is on offer here is a solution for the near-term future rather than something that could be used in practice immediately. However, as has been mentioned, and is worth remembering, conventional computer graphics is as good as it is because of many years of research embedded today in fast hardware systems. Is it worth continuing with the familiar paradigm, or investing in this new approach that provides a way out of many of the entanglements of current day graphics processing?

This new method does require a new way of thinking about graphics which will seem strange to many working in the field: there is no need for a graphics pipeline; depth of field comes free; there is no need for level of detail or visibility culling; there is no more time involved displaying a scene with 1 object compared to 1 million objects; objects are not the central focus of computer graphics rendering; objects do not have to be modelled as polygons; objects are not rendered onto the display in any direct manner.

The major problem with the new method is the need for a very large amount of memory to store the VLF. This problem is likely to be alleviated through the use of a compression scheme - discussed in some detail by Levoy and Hanrahan (1996). Other issues not covered in this paper are filtering to deal with high frequency in the radiance, and the extraction of a pin-hole camera type of view from the VLF. These are also discussed in (Levoy and Hanrahan, 1996; Gortler, et. al., 1996) and methods in those papers could be adapted here. The paper has clearly assumed throughout the transport of light in a non-participating medium (in fact a vacuum as an approximation to air).

References

- Arvo, J. and Kirk, D. (1987) Fast Ray Tracing by Ray Classification, *Computer Graphics* 21(4), 55-64.
- Arvo, J. and Kirk, D. (1990) Particle Transport and Image Synthesis, *Computer Graphics (SIGGRAPH)*, 24(4), 63-66.
- Chapman, J. Calvert, T.W., Dill, J. (1991) Spatio-Temporal Coherence in Ray Tracing, *Graphics Interface '91*, 101-108.
- Chen, E. (1990) Incremental Radiosity: An Extension of Progressive Radiosity to an Interactive Image Synthesis System, *Computer Graphics* 24(4), 135-144.
- Chrysanthou, Y., Daniel Cohen-Or and Dani Lischinski (1998) Fast Approximate Quantitative Visibility for Complex Scenes, *Computer Graphics International '98*, Hannover, Germany, June 1998, 220-227.

- Clark, J.H. (1976) Hierarchical Geometric Models for visible surface algorithms, *Communications of the ACM*, 19(10), 547-554.
- Cleary, J.G., Wyvill, G. (1988) Analysis of an Algorithm for Fast Ray Tracing Using Uniform Space Subdivision, *The Visual Computer*, 4, 65-83.
- Cohen, M.F., Greenberg, D.P. (1985) The Hemi-Cube: A Radiosity Solution for Complex Environments, *Computer Graphics* 19(3), 31-40.
- Cohen, M.F., Shenchang, Ec., Wallace, J.R. and Greenberg, D.P. (1988) A Progressive Refinement Approach to Fast Radiosity Image Generation, *Computer Graphics* 22(4), 75-84.
- Cook, R.L., Porter, T., Carpenter, L. (1984) Distributed Ray Tracing, *Computer Graphics (SIGGRAPH)* 18(3), 127-146.
- Foley, J.D., van Dam, A., Feiner, S.K., Hughes, J.F. (1990) *Computer Graphics: Principles and Practice*, Addison-Wesley, Reading M.A., 2nd ed..
- Fujimoto, A., Tanaka, T., Iwata, K. (1986) ARTS: Accelerated Ray-Tracing System, *IEEE CG&A* 6(4), 16-26.
- Glassner, A.S. (1984) Space Subdivision for Fast Ray Tracing, *IEEE Computer Graphics and Applications*, 4(10), 15-22.
- Goral, C.M., Torrance, K.E., Greenburg, D.P. (1984) Modeling the Interaction of Light Between Diffuse Surfaces, *Computer Graphics (SIGGRAPH)* 18(3), 213-222.
- Gortler, S., Grzeszczuk, R., Szeliski, R., Cohen, M. (1996) The Lumigraph, *Computer Graphics (SIGGRAPH)*, Annual Conference Series, 43-52.
- Gu, X., Gortler, S.J., Cohen, M. (1997) Polyhedral Geometry and the Two-Plane Parameterization, *Rendering Techniques '97*, Ed. J. Dorsey and P. Slusallek, Eurographics, Springer, 1-12.
- Hanrahan, P. et. al. (1991) A Rapid Hierarchical Radiosity Algorithm, *Computer Graphics (SIGGRAPH)* 25(4) 197-206.
- Kajiya, J.T. (1986) The Rendering Equation, *Computer Graphics (SIGGRAPH)*, 20(4), 143-150.
- Kaufman, A.E. (1996) Volume Synthesis, in *Discrete Geometry for Computer Imagery*, 6th International Workshop, DGCI'06, Lyon, France, Proceedings, ed. S. Miquet, A. Montanvert, S. Ubeda, Springer.
- Levoy M, Hanrahan, P. (1996) Light Field Rendering, *Computer Graphics (SIGGRAPH)*, Annual Conference Series, 31-42.

Muller, H., Winckler, J. (1992) Distributed Image Synthesis with Breadth-First Ray Tracing and the Ray-z-Buffer, Data Structures and Efficient Algorithms. Final Report on the DFG Special Initiative, in B. Monien and T. Ottmann eds. Lecture Notes in Computer Science, 594, 124-147, Springer-Verlag.

Nakamaru, K., Ohno, Y. (1997) Breadth-First Ray Tracing Utilizing Uniform Spatial Subdivision, IEEE Transactions on Visualization and Computer Graphics, 3(4), 316-328.

Rohlf, J. and Helman, J. (1994) IRIS Performer: A High Performance Multiprocessing Toolkit for Real-Time 3D Graphics, Computer Graphics (SIGGRAPH Proceedings), 381-394.

Shirley, P., Wade, B., Zareski, D., Hubbard, P., Walter, B., Greenberg, D.P. (1995) Global Illumination via Density Estimation, Proceedings of the Sixth Eurographics Workshop on Rendering, June, 187-199.

Teller, S., Bala, K., Dorsey, J. (1996) Conservative Radiance Interpolants for Ray Tracing, Rendering Techniques '96, Proceedings of the Eurographics Workshop in Porto, Portugal, June 17-19, X. Pueyo and P. Schroder (eds), Springer, 257-268.

Wallace, J.R., Elmquist, K.A., Haines, E. (1989) A Ray Tracing Algorithm for Progressive Radiosity, 23(3), Computer Graphics 315-323.

Walter, B., Hubbard, P.M., Shirley, P., Greenberg, D.P. (1997) Global Illumination Using Local Linear Density Estimation, ACM Transactions on Graphics, 16(3), 217-259.

Whitted, T. (1980) An Improved Illumination Model for Shaded Display, Comm. ACM 24(6), 343-349, June.

Yagel, R., Cohen, D., Kaufman, A. (1992) Discrete Ray Tracing, IEEE Computer Graphics and Applications, 12(5), 19-28.

CLAIMS

1. Simulation apparatus operative to simulate propagation of energy within an environment, the apparatus comprising:

5 means for defining a plurality of paths along which energy can propagate in the environment; and

means for recording a point on one of said paths, at which point there is interaction within the environment with said path.

10

2. Apparatus in accordance with claim 1 and further comprising means for recording the propagation of energy along one of said paths from an emitter of energy within said environment, and means for identifying subsequent
15 propagation of said energy along at least one of said paths following consideration of the effect on said energy of an interaction with said path.

3. Apparatus in accordance with claim 2 wherein the
20 means for identifying includes means for tracking propagation of energy along a first path to a point of interaction on said path, means for finding a second path onto which energy is to be simulated as propagating after consideration of the interaction, and means for
25 calculating the magnitude of energy propagating along said second path after consideration of the interaction.

4. Apparatus in accordance with any preceding claim wherein said means for defining a plurality of discrete paths is operative to define a plurality of discrete paths consisting of a plurality of sub-sets of parallel
5 discrete paths.

5. Apparatus in accordance with claim 4 wherein, in use, each sub-set comprises discrete paths in a direction different from the direction of all other discrete paths.
10

6. Apparatus in accordance with any one of claims 1 to 3 wherein ends of said discrete paths are defined in at least two planes, each plane containing a set of points, wherein one end of each discrete path is a member of said
15 set of points in one plane and another end of each discrete path is a member of said set of points in another plane.

7. Apparatus in accordance with claim 6 wherein the at
20 least two planes include at least one pair of parallel planes.

8. Apparatus in accordance with claim 6 or claim 7 wherein the at least two planes include at least one pair
25 of orthogonal planes.

9. Apparatus in accordance with any one of claims 6 to

8 wherein ends of said discrete paths are defined between
six planes enclosing a region in three-dimensional space.

10. Apparatus in accordance with claim 9 wherein said
5 region is substantially cuboidal.

11. Apparatus in accordance with any preceding claim
wherein said means for recording a point on one of said
paths includes means for recording the position of said
10 point on said path.

12. Apparatus in accordance with any preceding claim
wherein said means for recording a point on one of said
paths includes means for recording a direction of
15 potential propagation of energy along said path.

13. Apparatus in accordance with any preceding claim
wherein said means for recording a point on one of said
paths includes means for recording data representative
20 of magnitude and nature of energy propagated along said
one of said paths from said point.

14. Apparatus in accordance with any preceding claim
wherein said means for recording a point on one of said
25 paths includes means for recording identity of the object
in the environment with which the path interacts at that
point.

15. Apparatus in accordance with any preceding claim wherein the apparatus is operative to simulate propagation of light within an environment.

5

16. Apparatus in accordance with claim 15 wherein the apparatus is operative to simulate propagation of chromatic light.

10 17. Apparatus in accordance with any preceding claim including means for producing an image, said image being indicative of energy propagating within the environment.

15 18. Apparatus in accordance with claim 17 wherein the means for producing an image includes means for defining an aperture within the environment, the means for producing an image being operable to produce an image indicative of energy propagated along paths passing through said aperture.

20

19. A method of simulating propagation of energy through an environment, the method comprising the steps of:

defining a plurality of paths within the environment along which energy can propagate; and

25 recording at least one point on at least one of said paths, the or each point corresponding to an interaction in the environment with said path.

20. A method in accordance with claim 19 including the step of:

storing data relating to the or each point recorded
5 by said recording step in a data structure from which the data can be retrieved on demand.

21. A method in accordance with claim 20 wherein the storing step includes the step of loading said data into
10 a binary tree.

22. A method in accordance with any one of claims 19 to 21 wherein said recording step includes the steps of:

identifying in respect of the or each point recorded
15 the potential direction of energy propagation from said point; and

identifying in respect of the or each point recorded the object within the environment with which the path containing said point interacts at that point.

20

23. A method in accordance with claim 22 wherein said recording step further includes the step of:

identifying in respect of the or each point recorded a value of energy propagated along said path from said
25 point in said identified direction.

24. A method in accordance with claim 23 and further

including the step of propagating energy along paths amongst the plurality of discrete paths, the step of propagating including the steps of:

5 identifying a source of energy within the environment;

finding points, amongst points recorded by the recording step, of the source with paths amongst the plurality of paths;

10 for each point found by said finding step, setting said value of energy propagated from said point in accordance with an energy emission characteristic of said source.

25. A method in accordance with claim 24 and further including the steps of:

15 for each point for which said value of energy is set, identifying the path containing said point and finding a further point, if any, further along said path in said identified direction, and marking the object associated with that further point as being in receipt of energy.

26. A method in accordance with claim 25 and further including the steps of:

25 for the or each object marked as being in receipt of energy, finding the or each further path along which said energy is to propagate following interaction at the

point, and setting a value of energy along the or each said further paths in accordance with the nature of the interaction.

5 27. A storage medium comprising a set of processor implementable instructions for performing the method of any one of claims 19 to 26.

10 28. A signal carrying a set of processor implementable instructions for performing the method of any one of claims 19 to 26.

15 29. A storage medium comprising a set of data defining a plurality of paths having associated therewith records of points on said paths at which exist interactions with objects of an environment.

20 30. A storage medium in accordance with claim 29 wherein said records include records of energy propagation along portions of said paths between said points of interactions.

25 31. A signal carrying a set of data defining a plurality of paths having associated therewith records of points in said paths at which exist interactions with objects of an environment.

32. A signal in accordance with claim 31 wherein said records include records of energy propagating along portions of said paths between said points of intersection.

5

33. A method of transmitting a signal comprising the steps of:

generating a signal in accordance with claim 31 or claim 32; and

10 transmitting said signal.

34. A method of constructing a graphical image, the method comprising:

15 receiving data defining a plurality of discrete paths having associated therewith records of points on said paths at which exist interactions with objects of an environment;

defining a viewer aperture within the plurality of paths;

20 defining a viewer image plane in said plurality of paths;

constructing an image defined by points of intersection of paths passing through the aperture with the image plane.

25

35. A method of manufacturing a storage medium including data defining a plurality of paths having associated

therewith records of points on said paths at which exist interactions with objects of an environment, the method comprising the steps of:

conducting the method in accordance with any one of
5 claims 19 to 26;

storing data defining said plurality of paths on a storage medium; and

storing data defining records of points on said paths on said storage medium.

10

36. A virtual reality system comprising:

apparatus in accordance with any one of claims 1 to 16; and

left and right image generation means, each image
15 generation means being operable to produce an image from a viewpoint within the environment of energy propagated towards the viewpoint.

37. A virtual reality system in accordance with claim
20 36 and further including left and right display means for displaying said images for stereoscopic viewing thereof.

38. Apparatus for monitoring light distribution through an environment comprising apparatus in accordance with
25 claim 15 or claim 16 and light measuring means operable to define a region of the environment and to measure light propagating through said region.

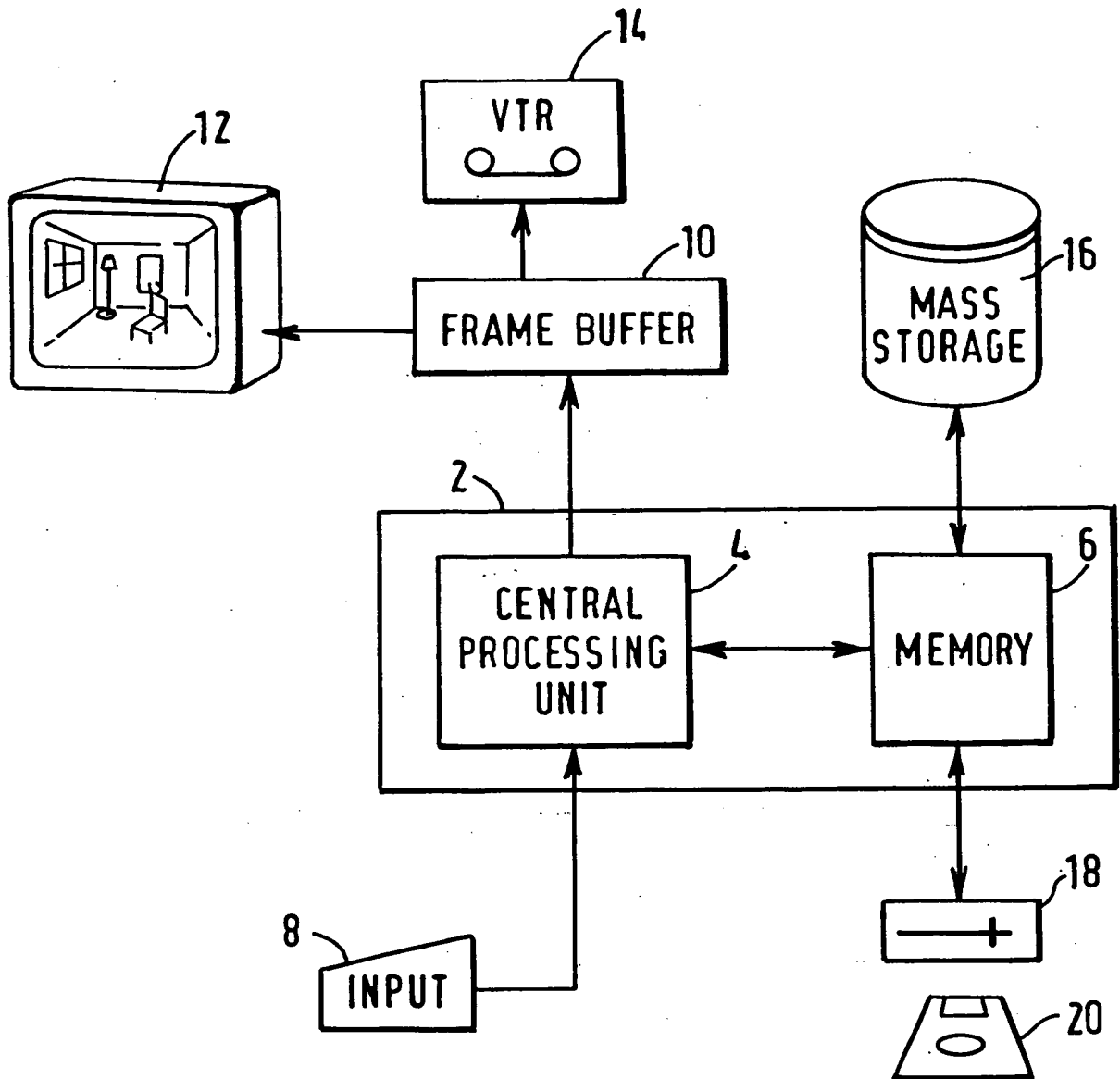
ABSTRACTENERGY PROPAGATION MODELLING

A simulation apparatus (2, 70) is operable to simulate
5 propagation of energy within an environment. The
apparatus comprises means for defining a plurality of
discrete paths (72) along which energy propagation in the
environment is to be traced, and means for recording a
point (t) on one of the paths, at which point there is
10 an interaction within the environment with the path (72),
A corresponding method is also provided.

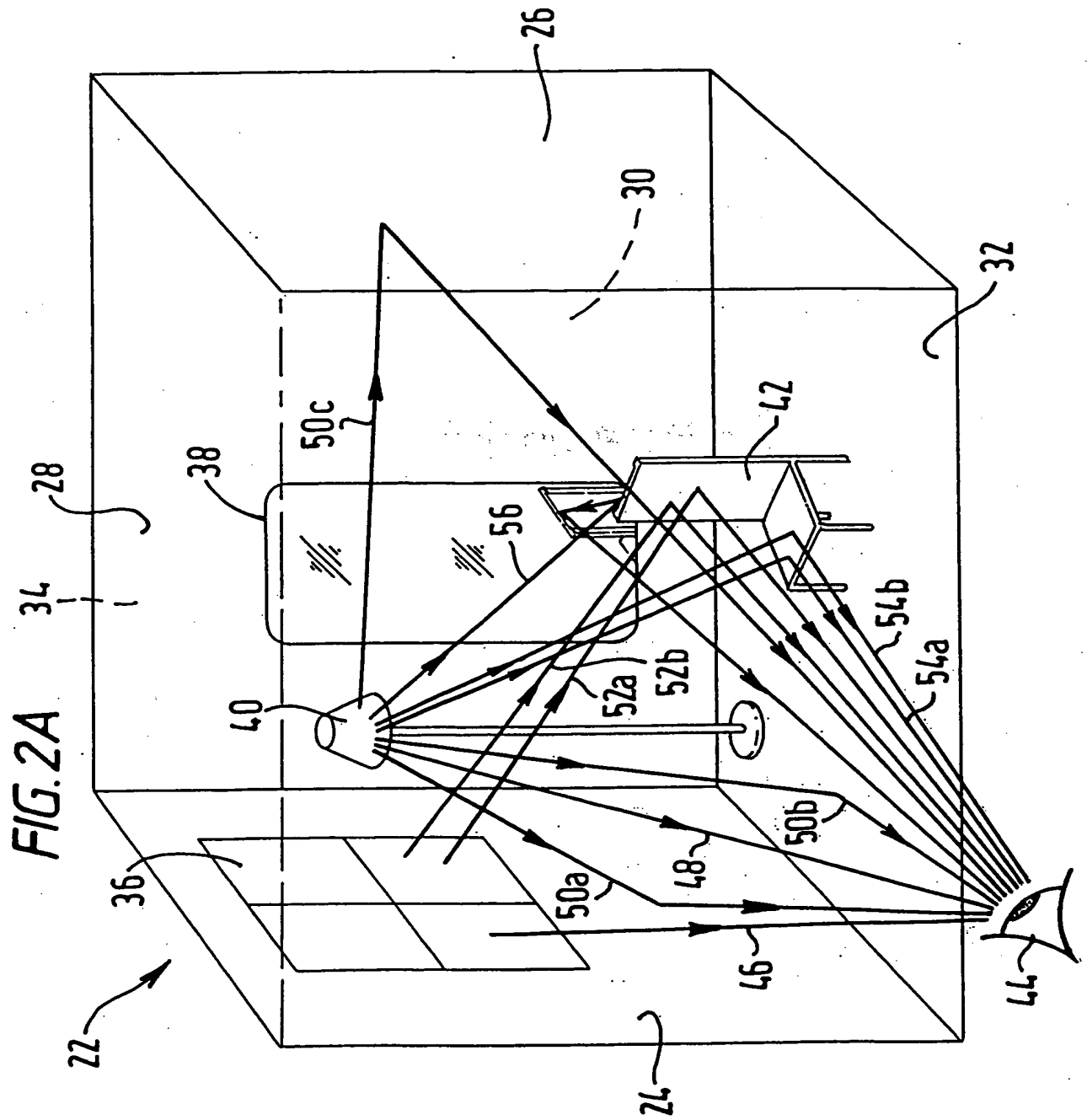
The invention further provides apparatus (2) for
monitoring light distribution in an environment, and
15 apparatus (2, 254) for displaying a stereoscopic image,
for instance for use in virtual reality applications.



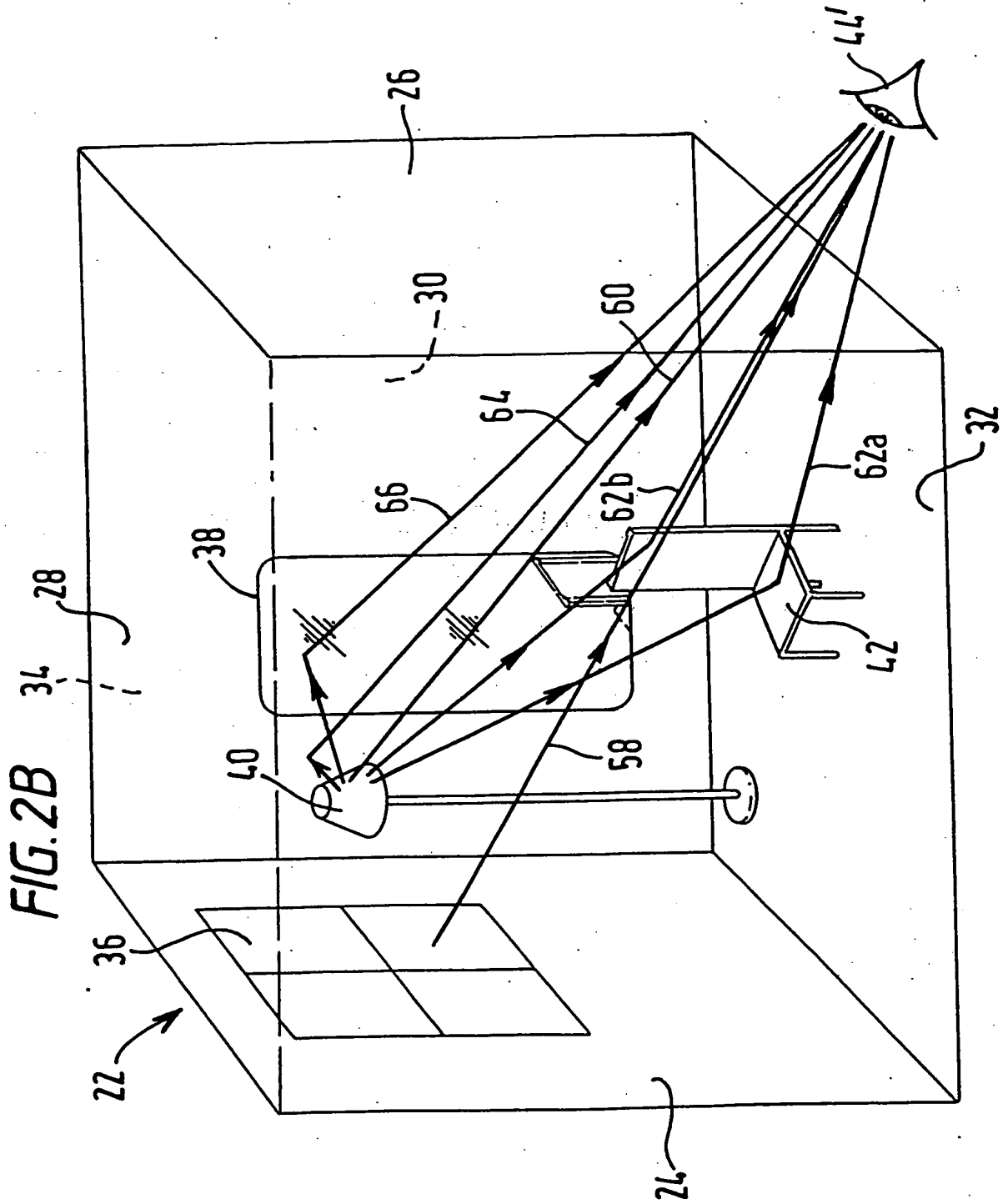
FIG. 1



This Page Blank (uspto)



This Page Blank (uspto)



This Page Blank (uspto)

FIG. 3A

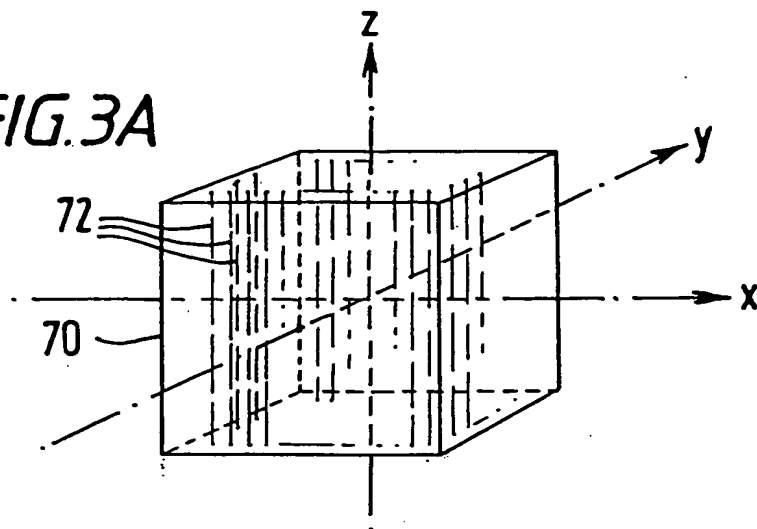


FIG. 3B

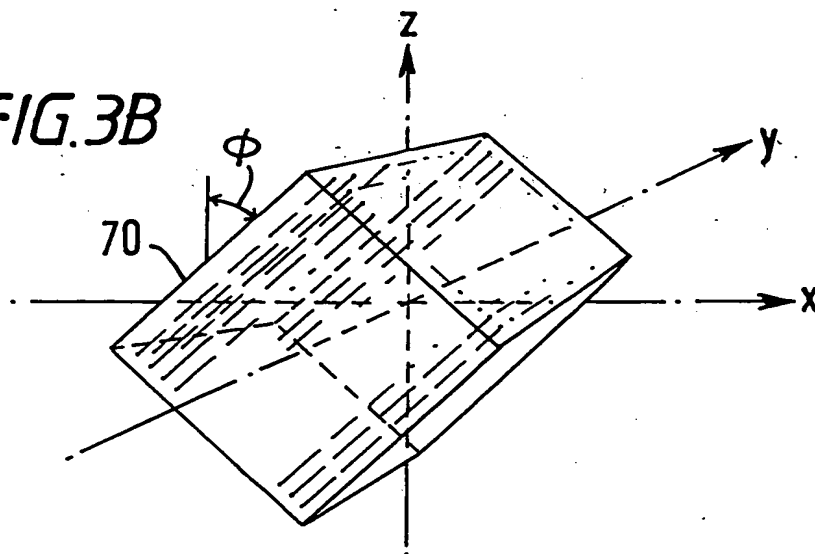
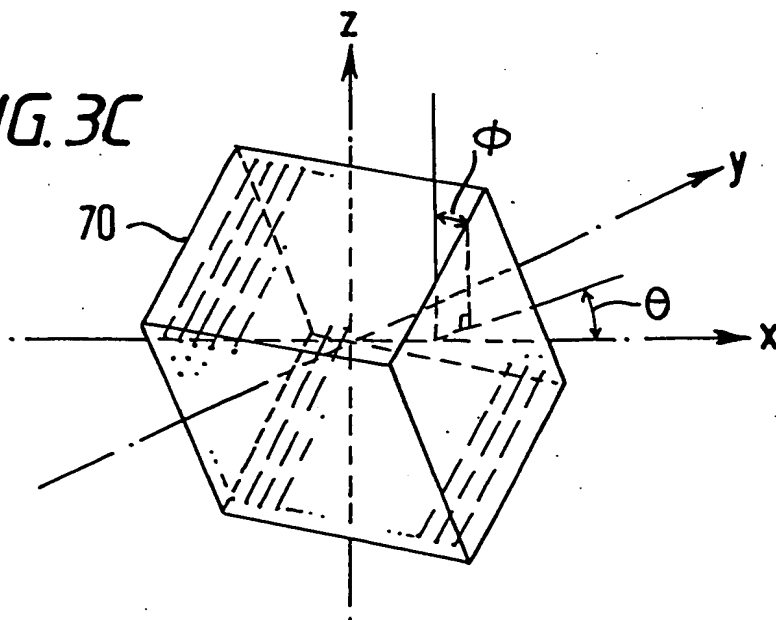
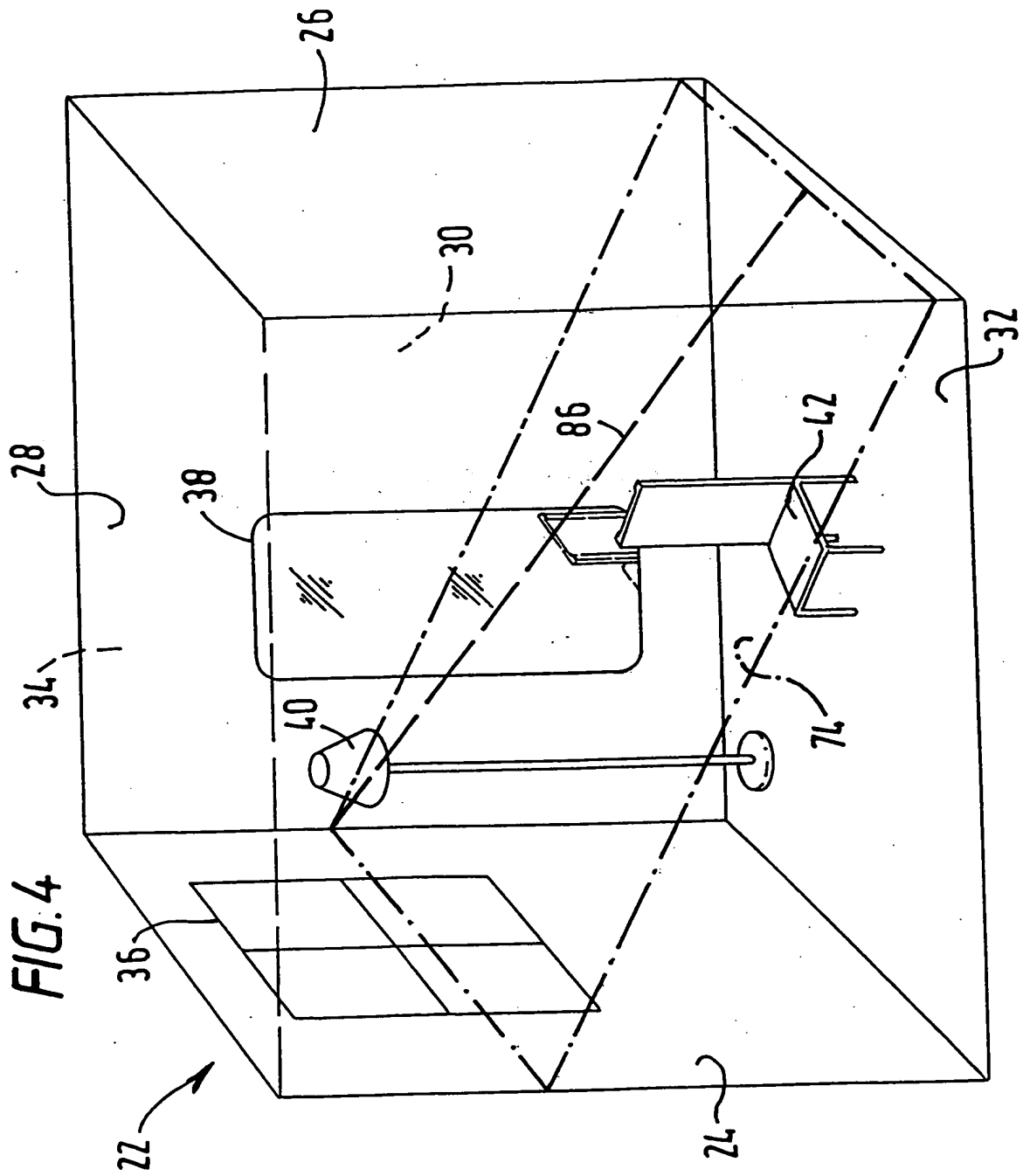


FIG. 3C

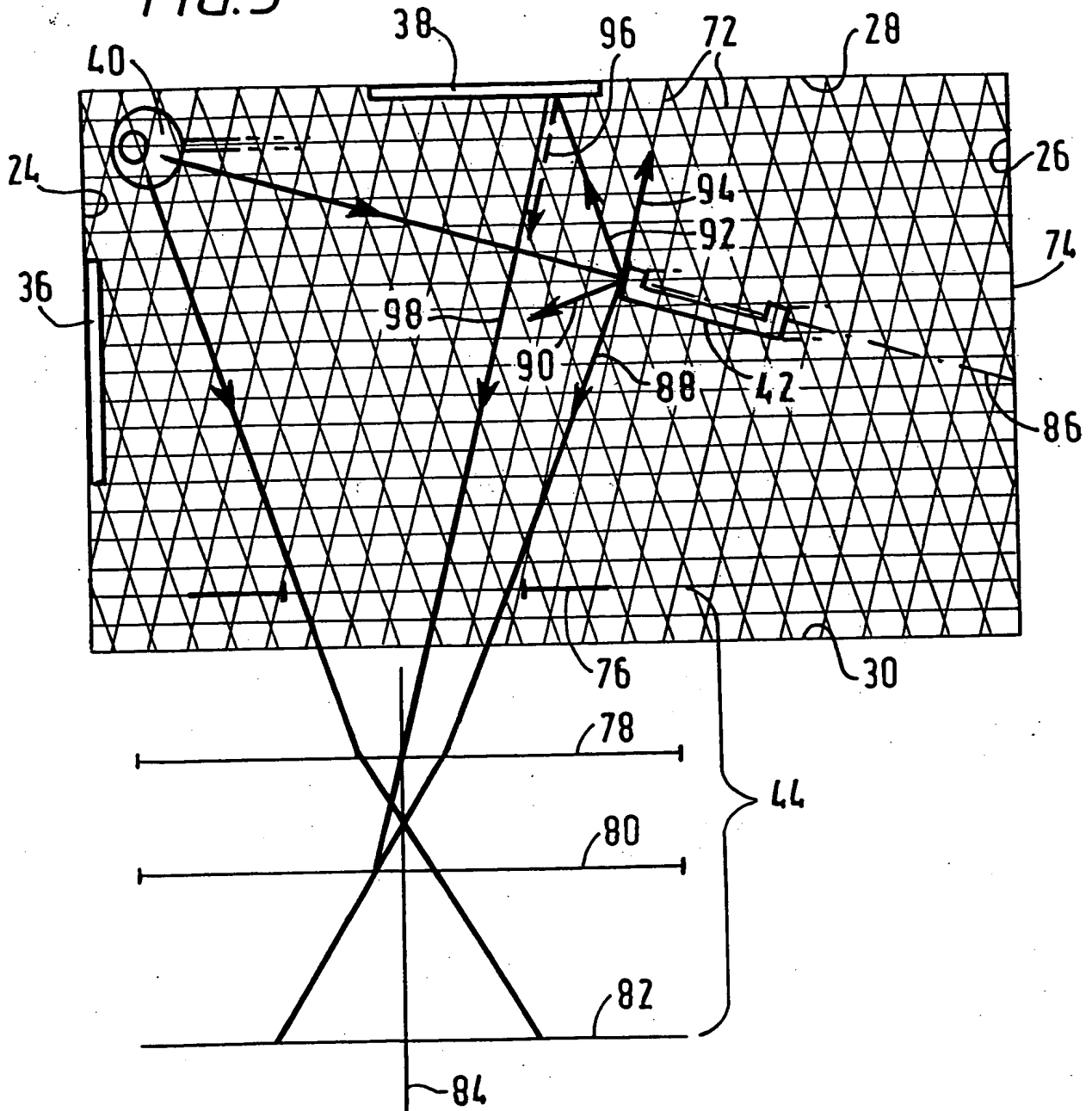


This Page Blank (uspto)



This Page Blank (uspto)

FIG. 5



This Page Blank (uspto)

7/32.

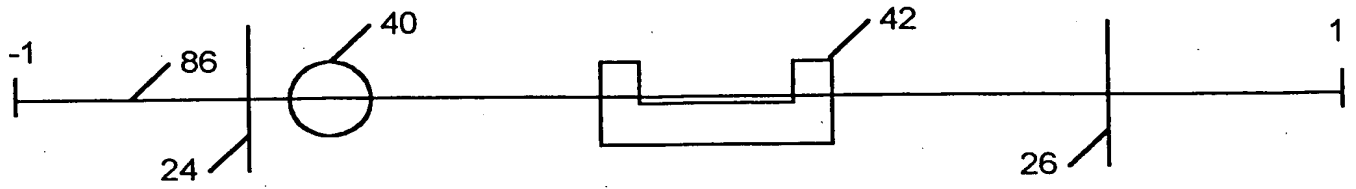


FIG. 6

This Page Blank (uspto)

8/32

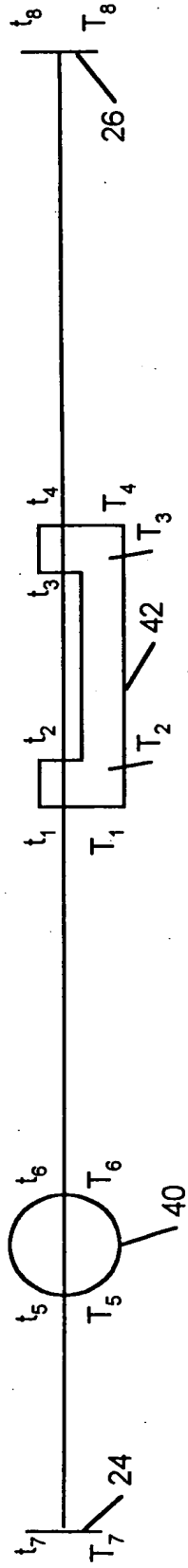


FIG. 7

This Page Blank (uspto)

9/32 .

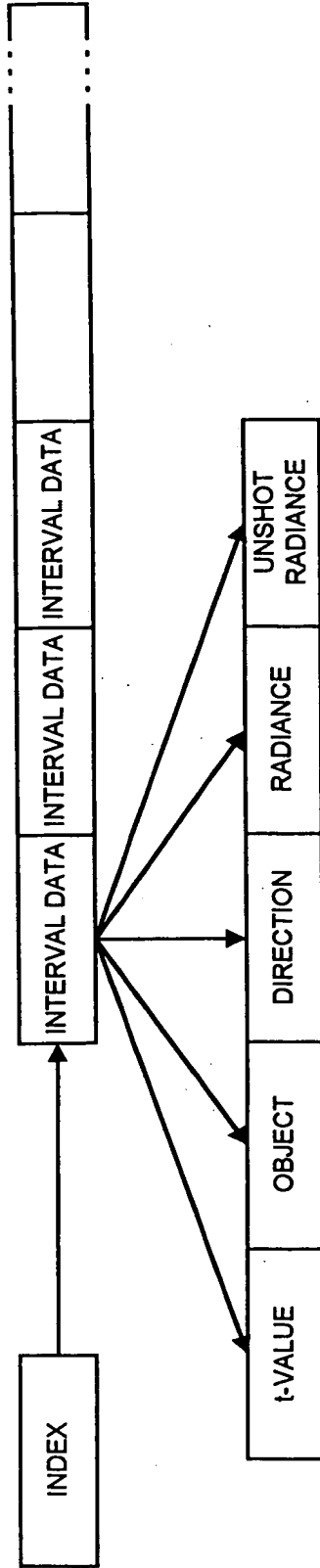


FIG. 8

This Page Blank (uspto)

10/32

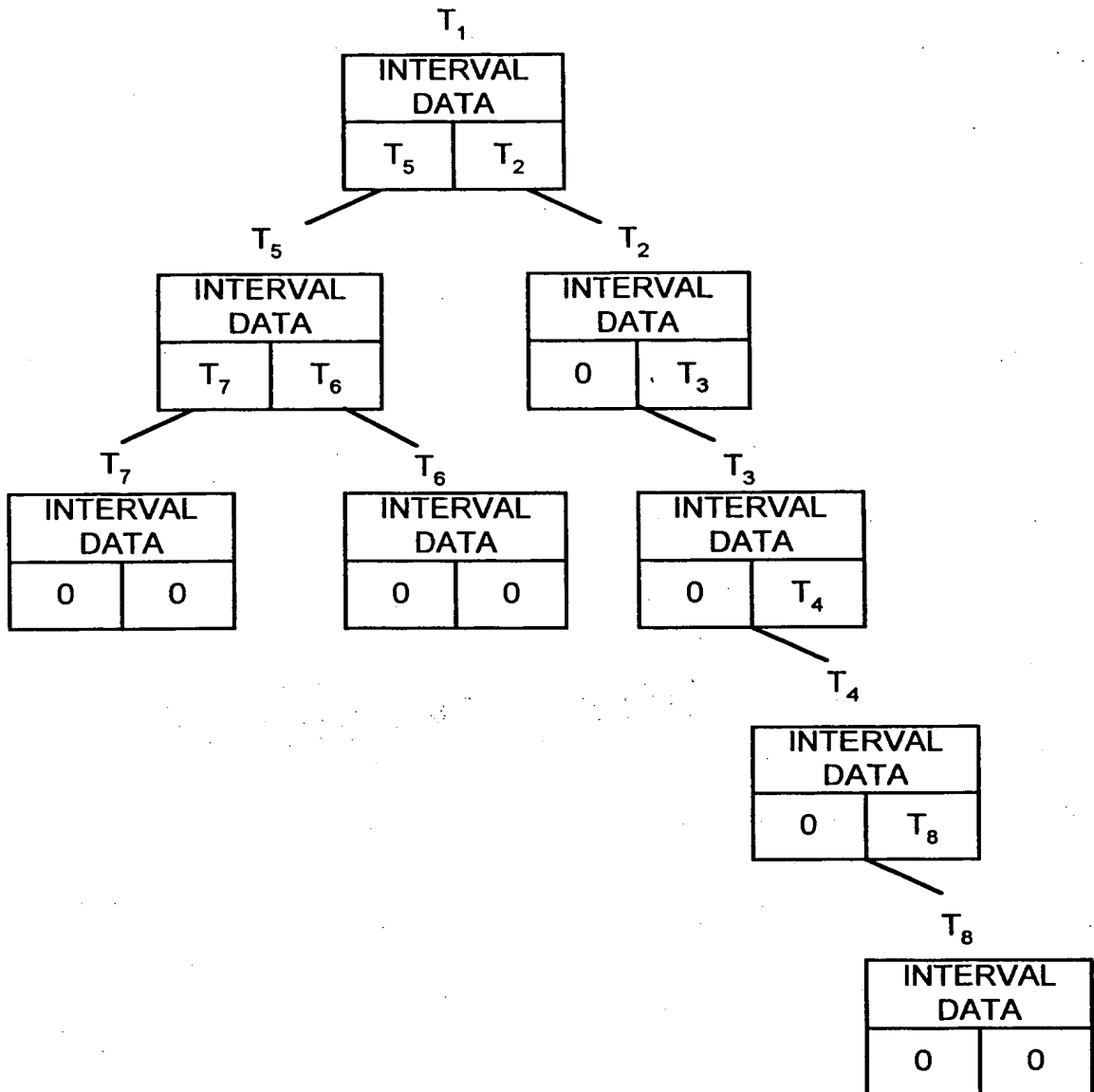


FIG. 9

This Page Blank (uspto)

11/32.

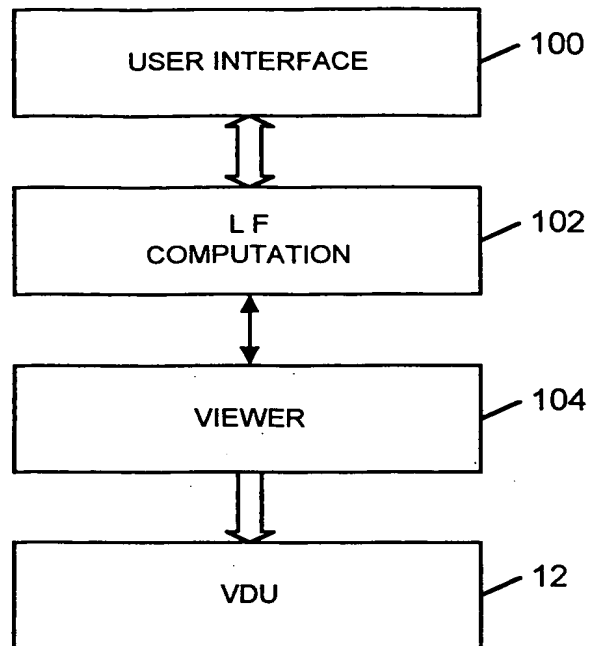


FIG. 10

This Page Blank (uspto)

12/32

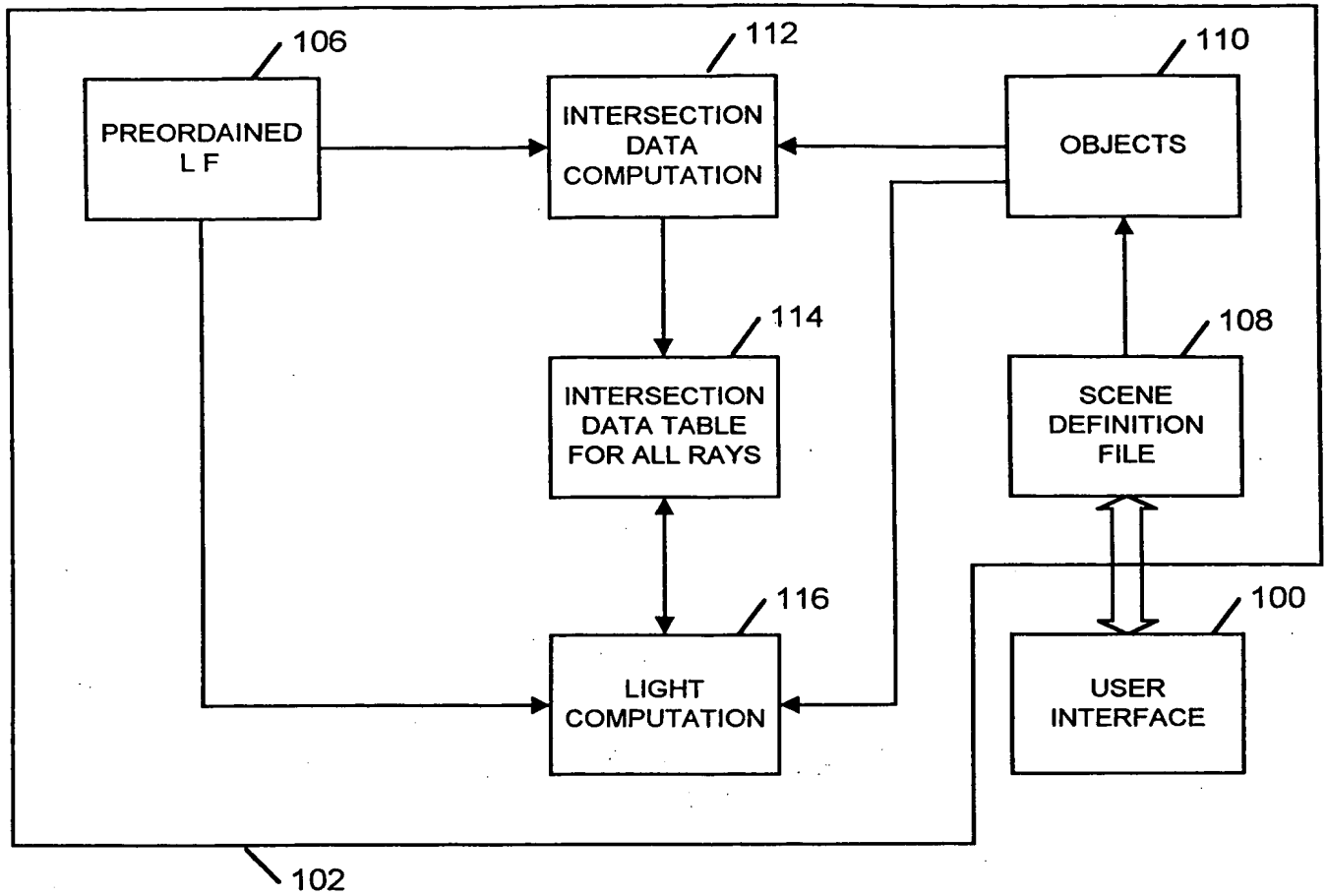


FIG. 11

This Page Blank (uspto)

13/32

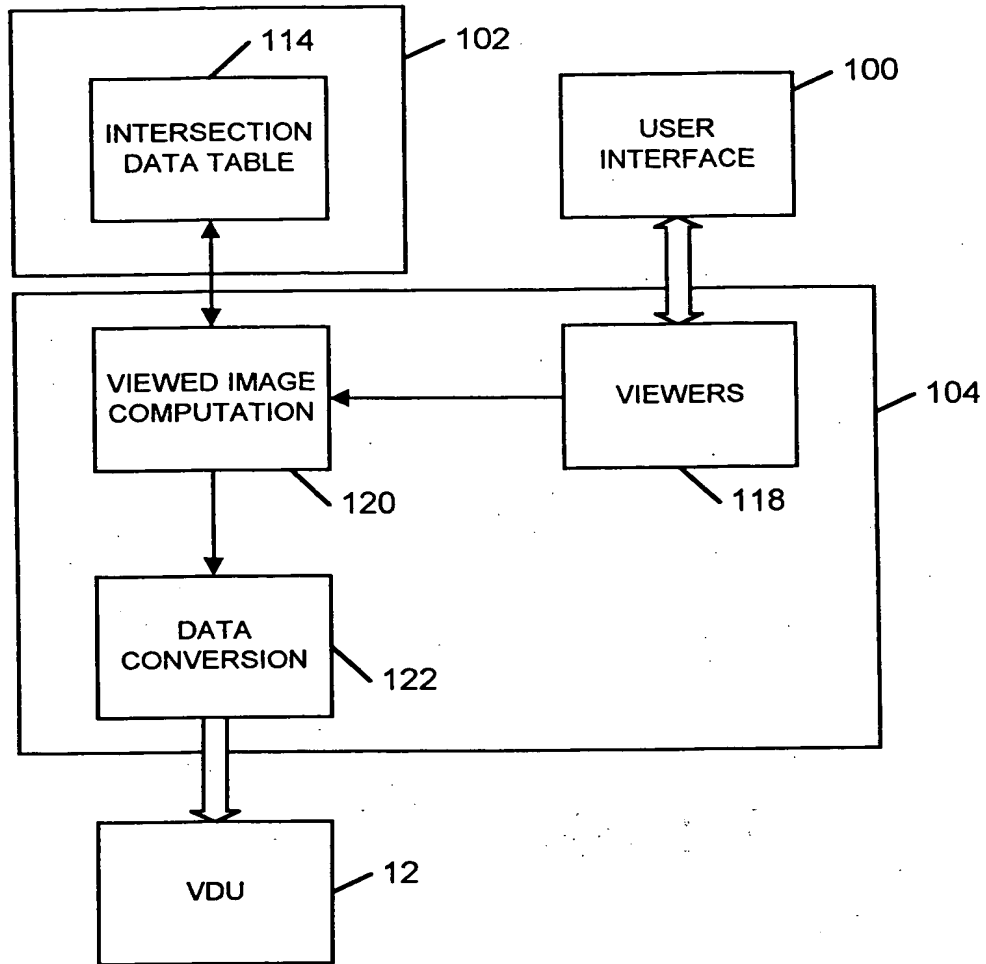


FIG. 12

This Page Blank (uspto)

14/32

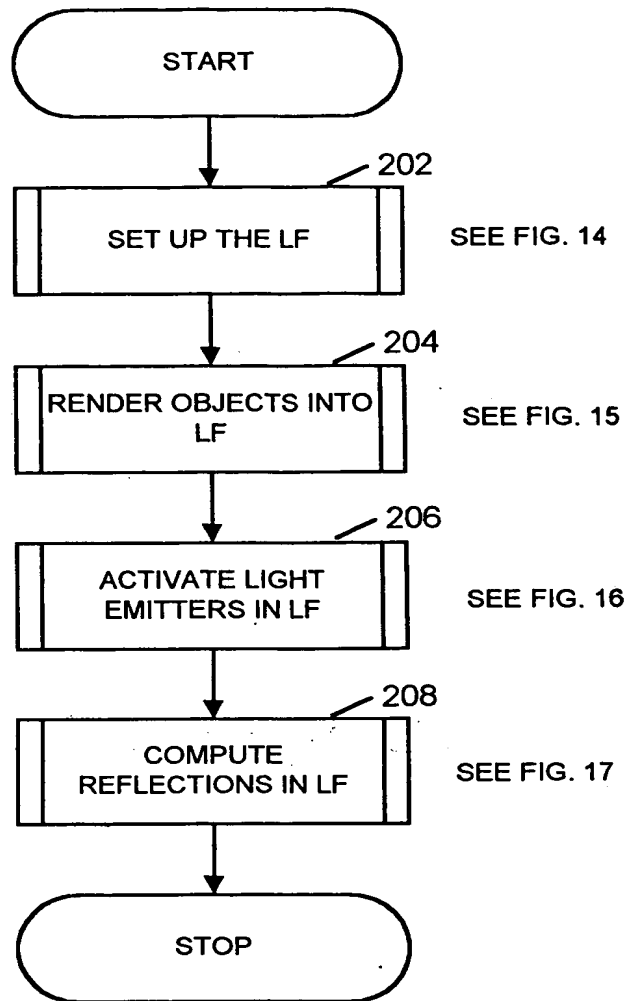


FIG. 13

This Page Blank (uspto)

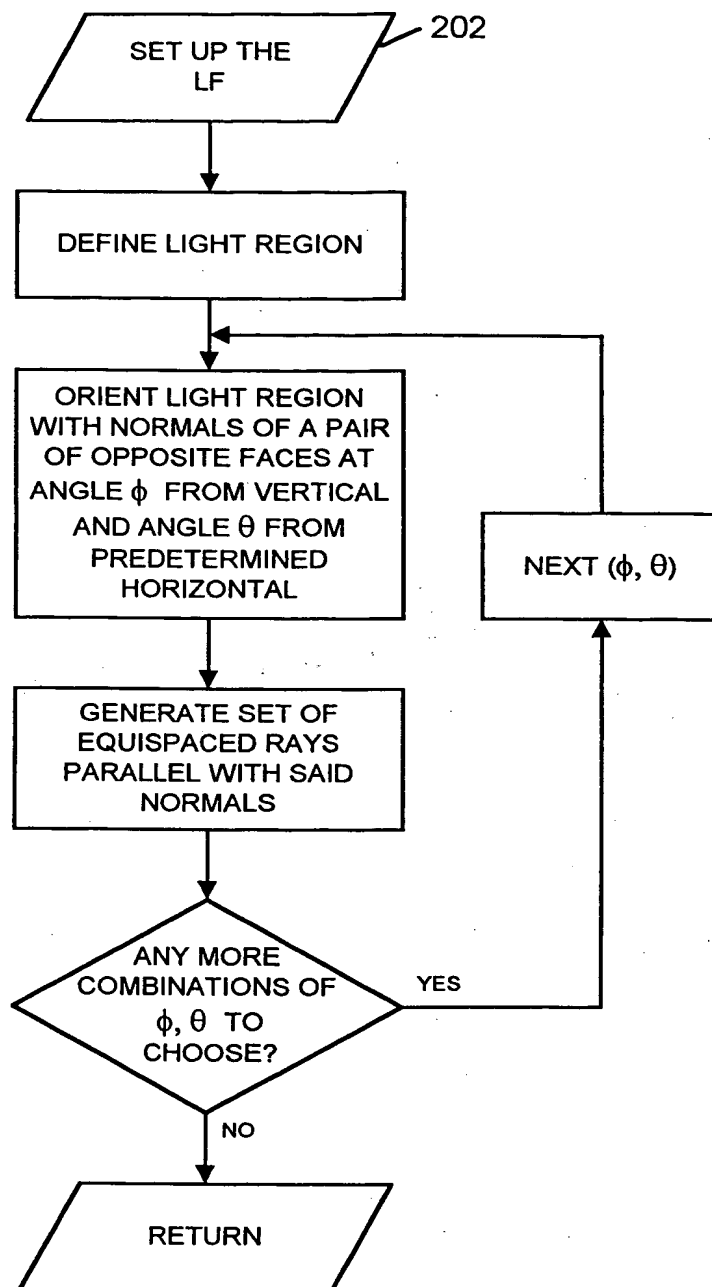


FIG. 14

This Page Blank (uspto)

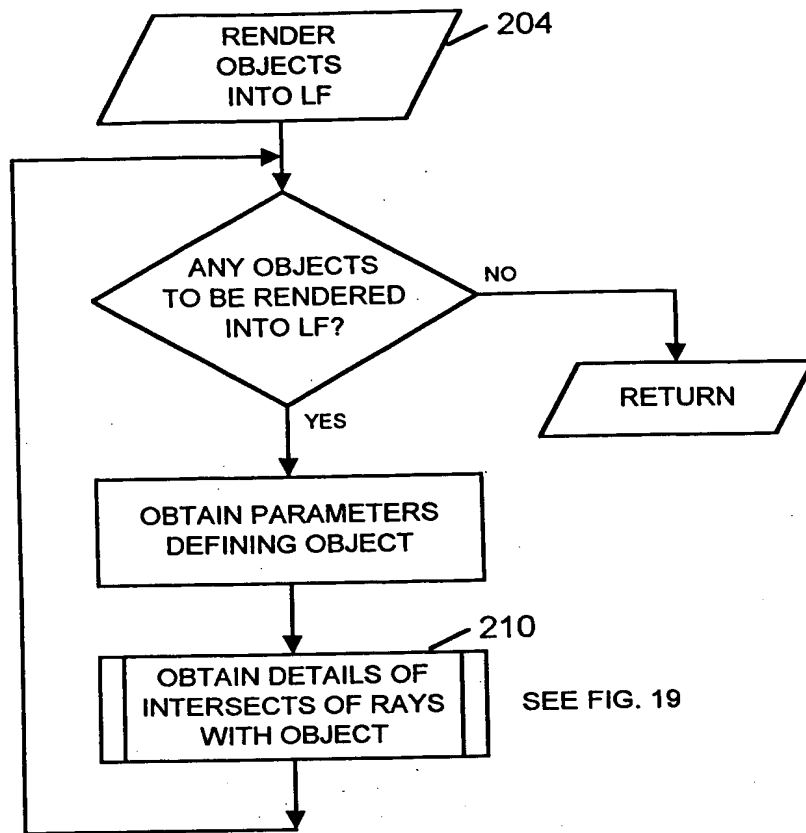


FIG. 15

This Page Blank (uspto)

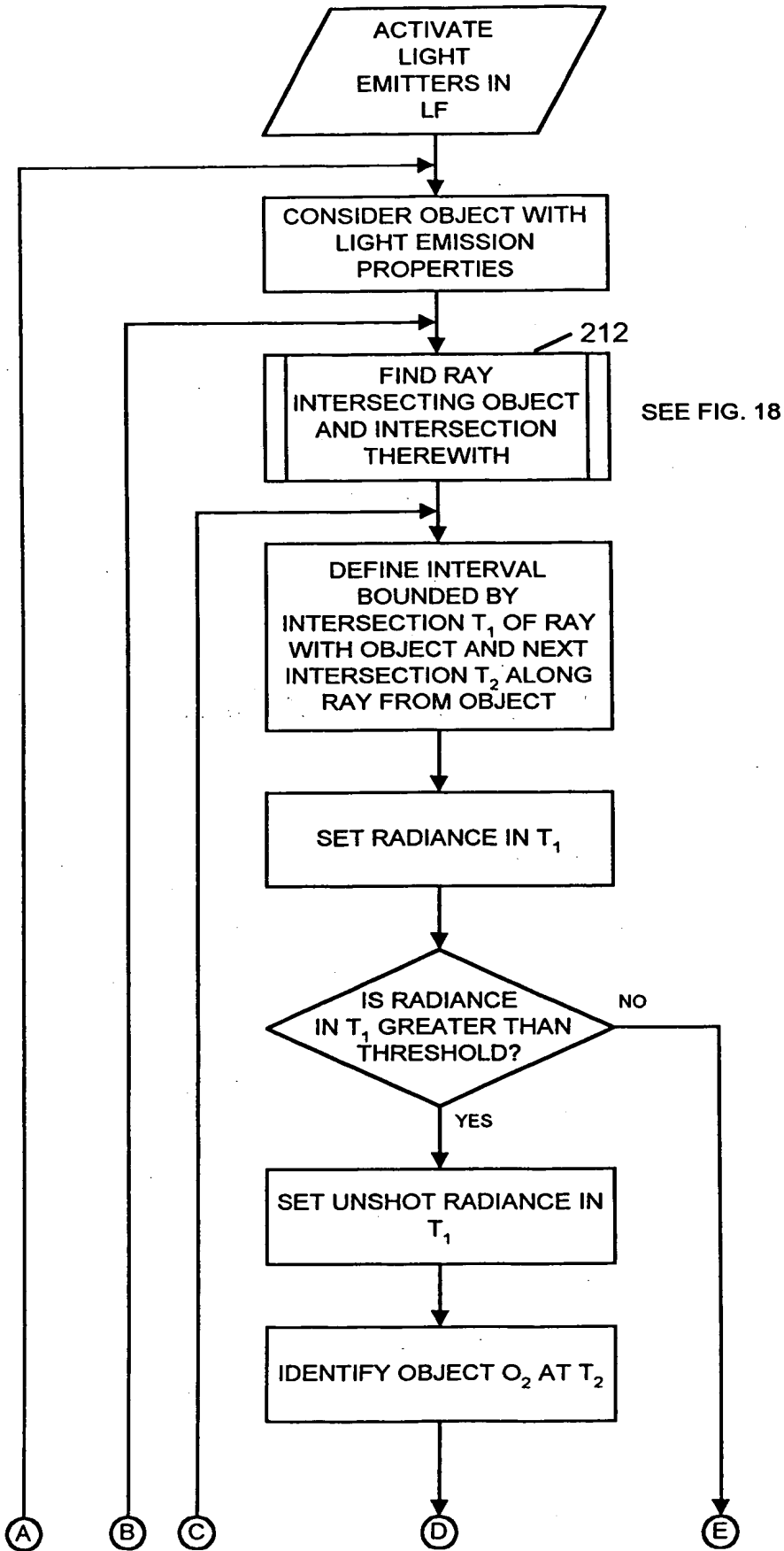


FIG. 16

This Page Blank (uspto)

18/32

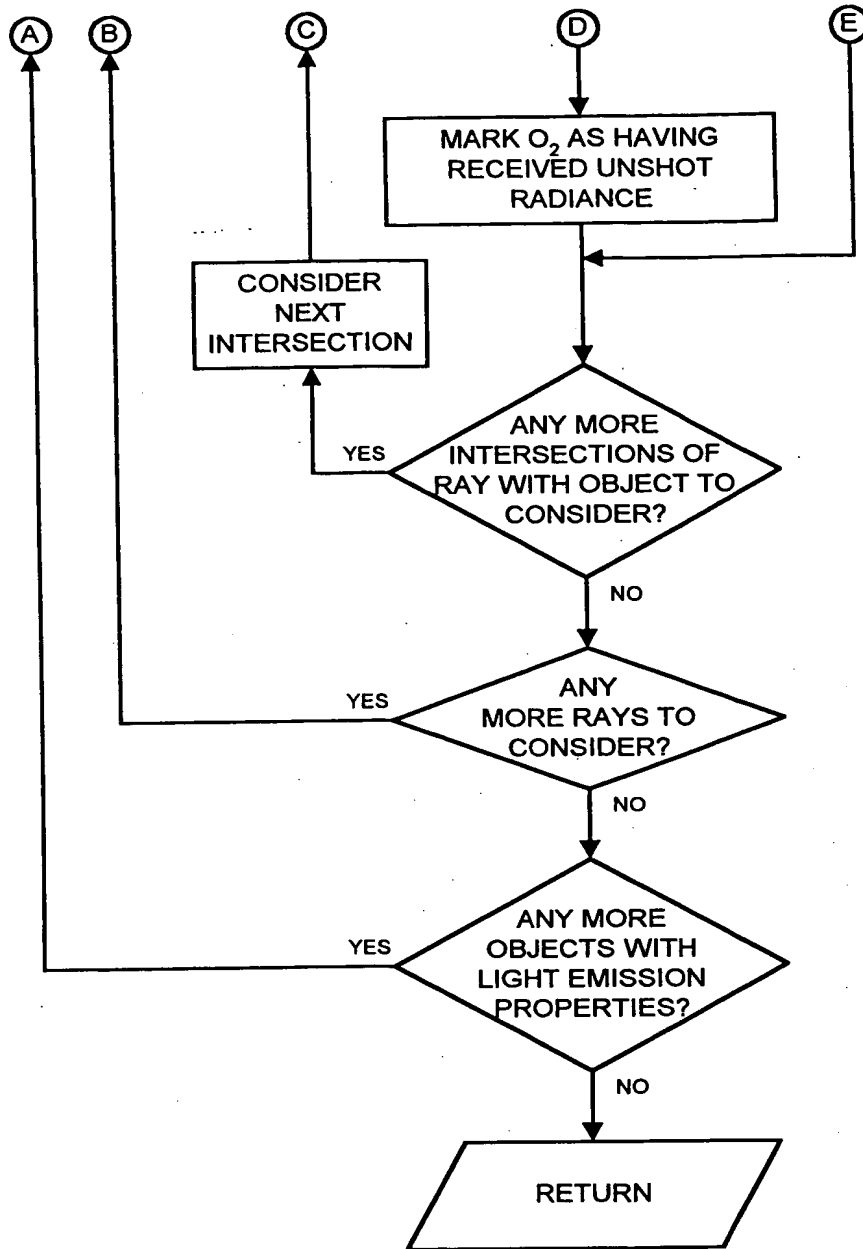


FIG. 16 (cont)

This Page Blank (uspto)

19/32

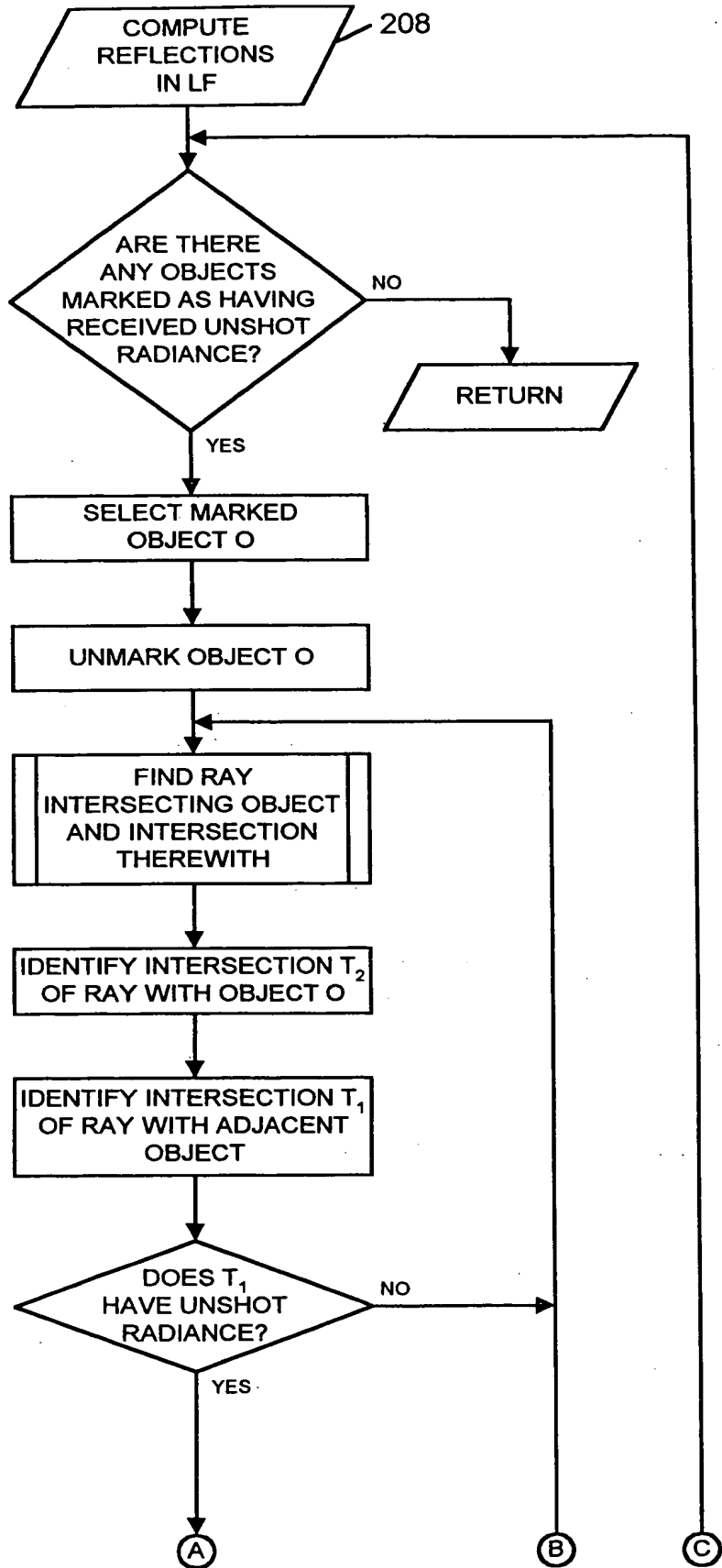


FIG. 17

This Page Blank (uspto)

20/32

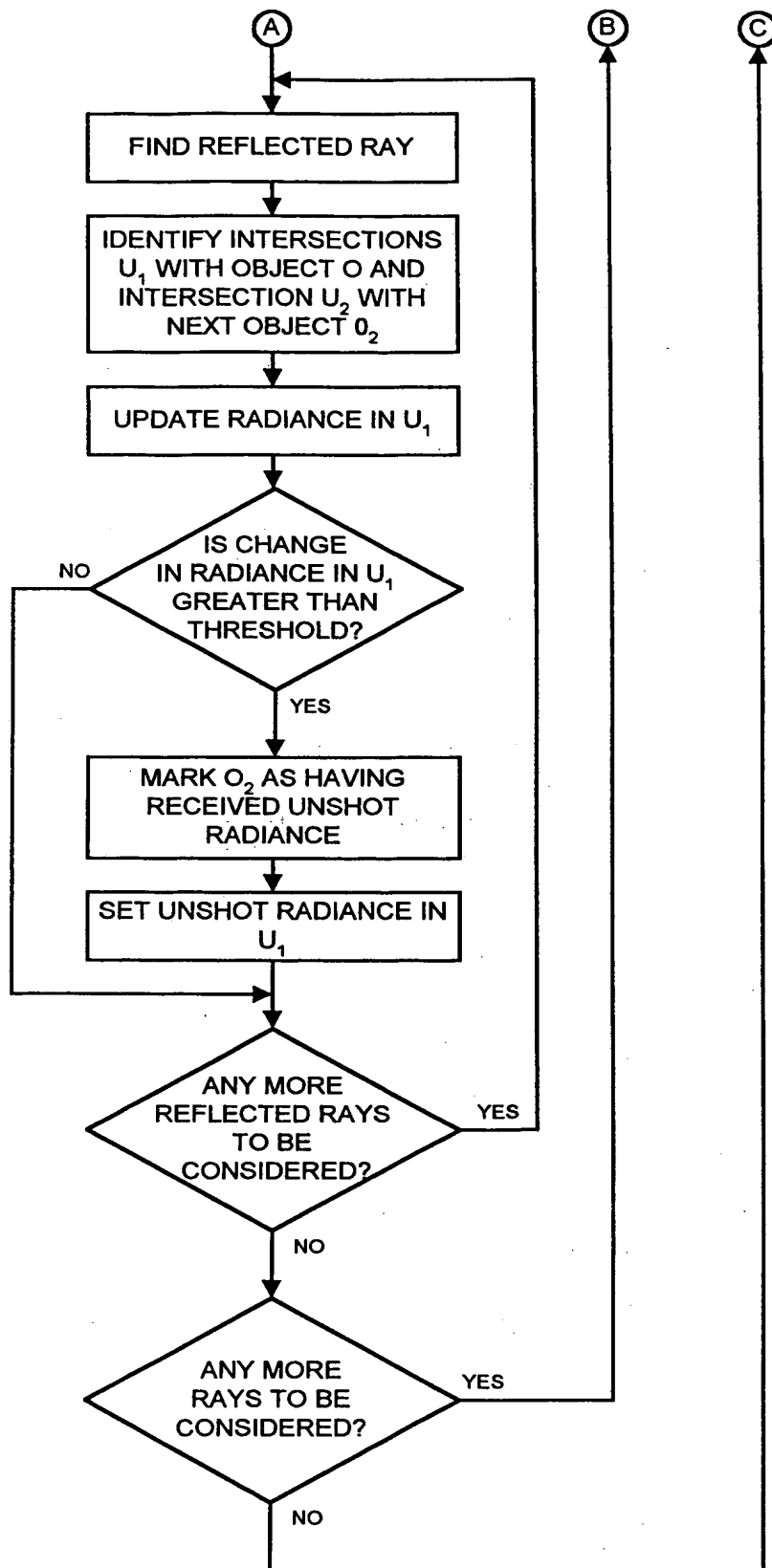


FIG. 17 (cont)

This Page Blank (uspto)

21/32.

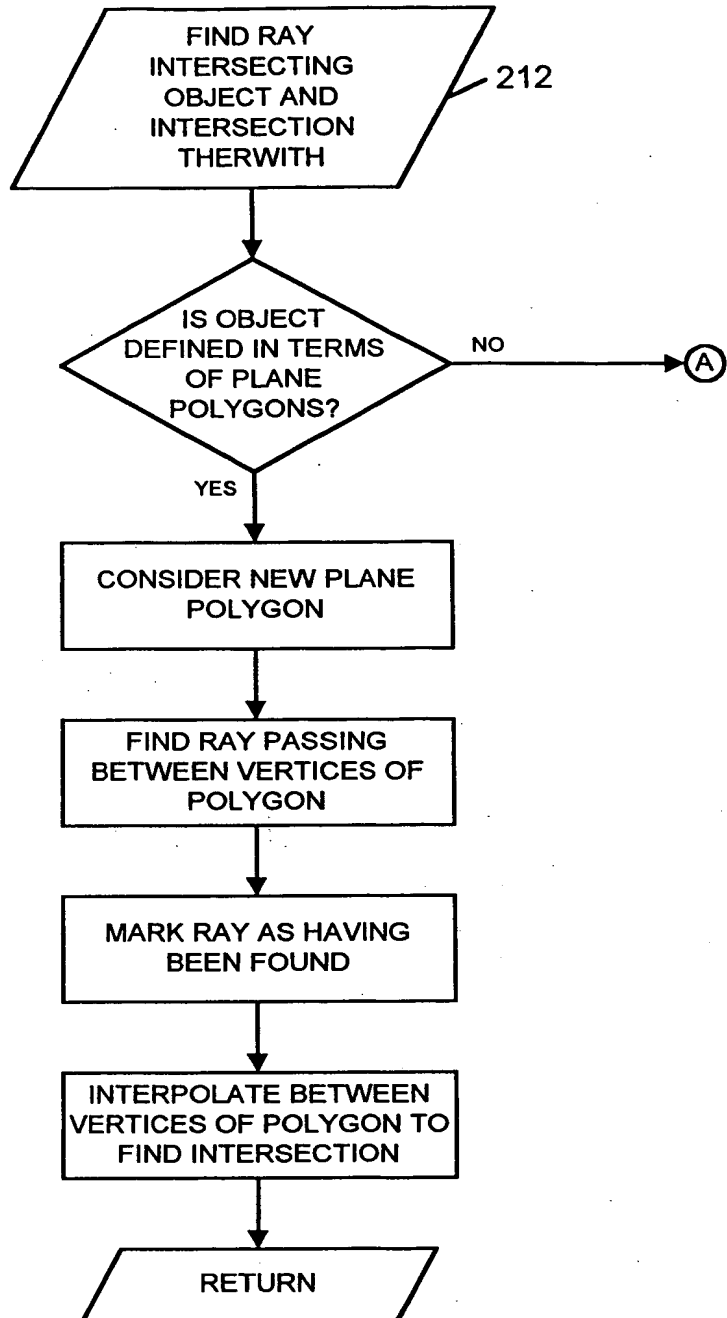


FIG. 18

This Page Blank (uspto)

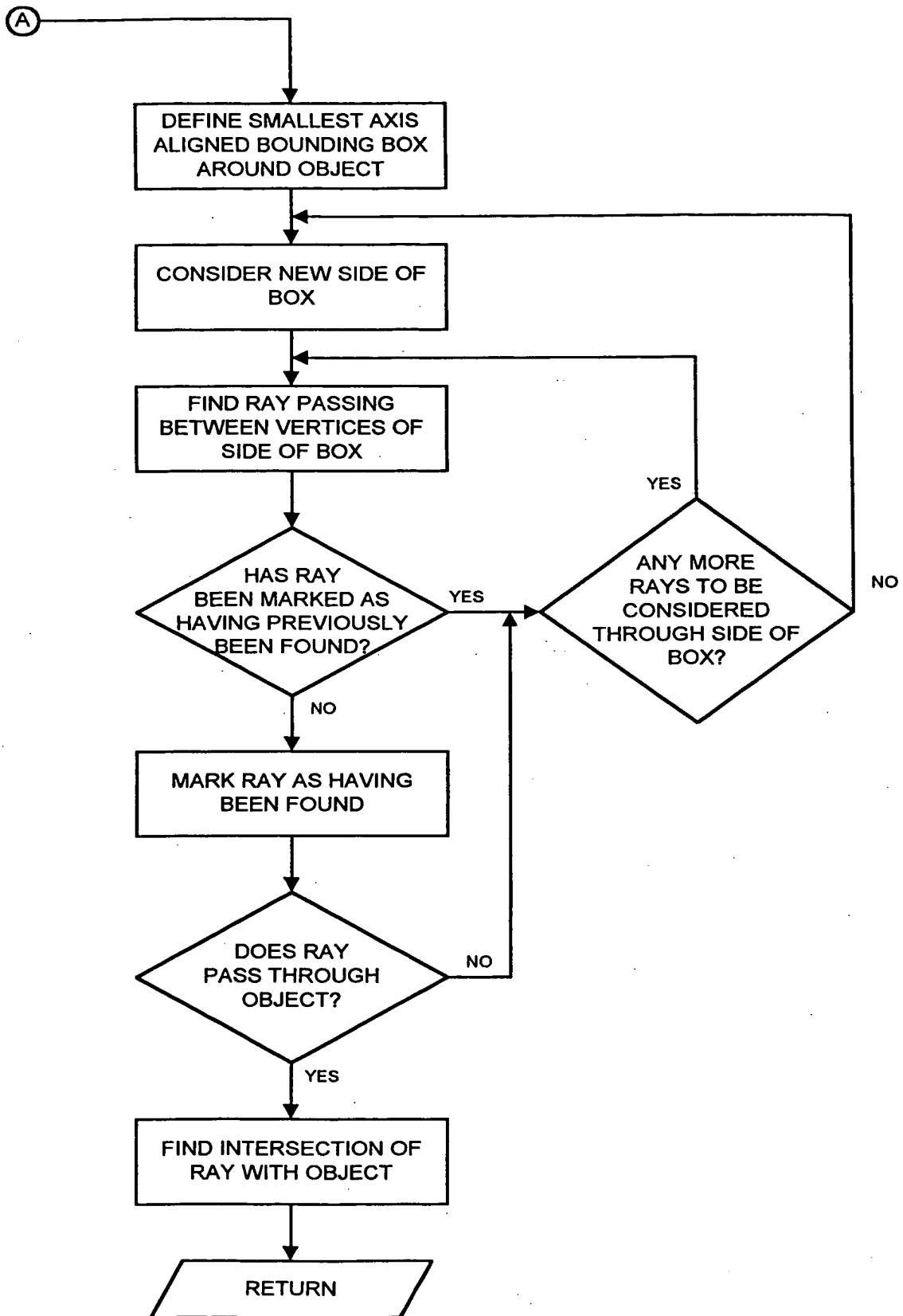


FIG. 18 (cont)

This Page Blank (uspto)

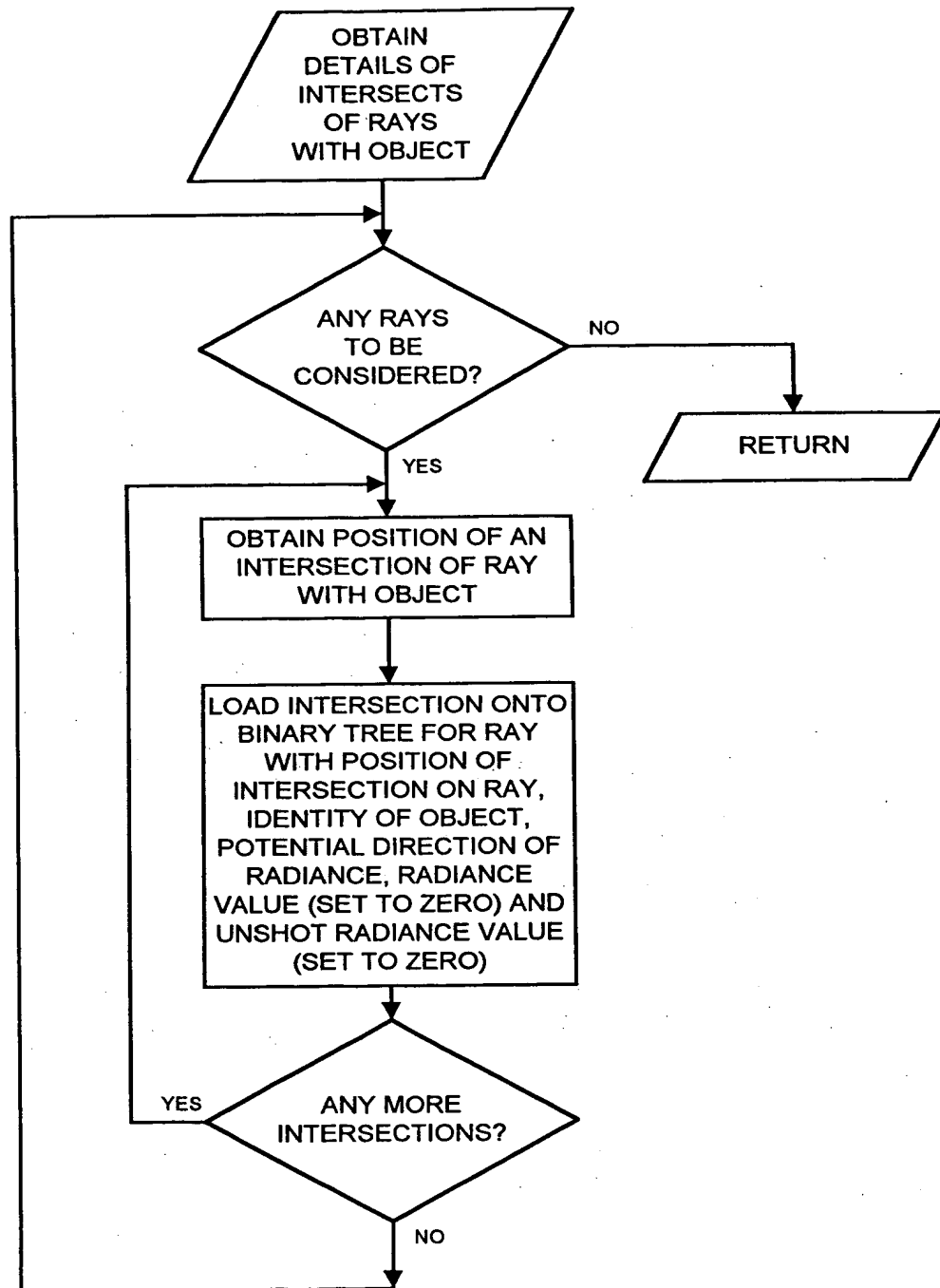


FIG. 19

This Page Blank (uspto)

24/32

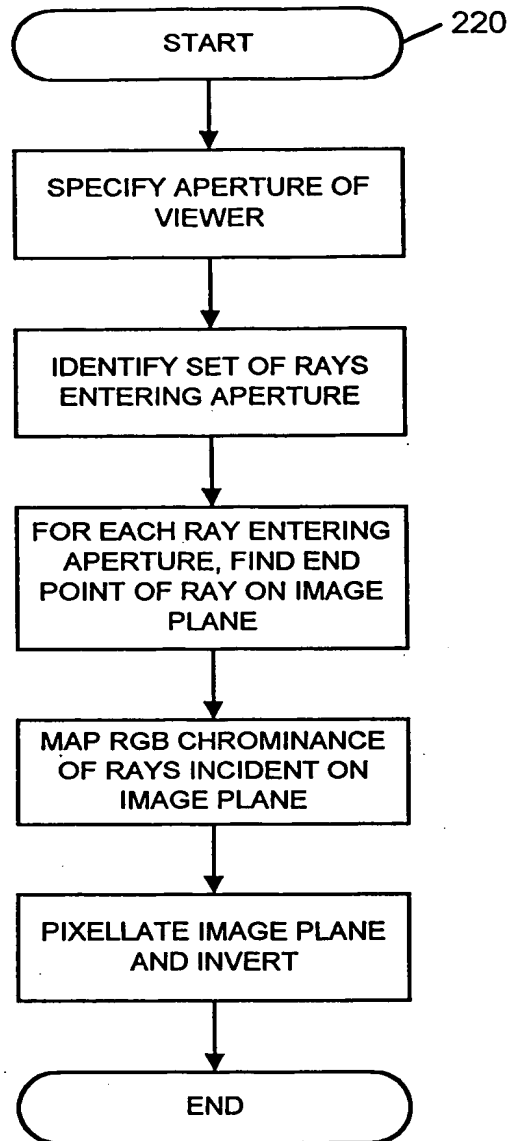


FIG. 20

This Page Blank (uspto)

25/32 .

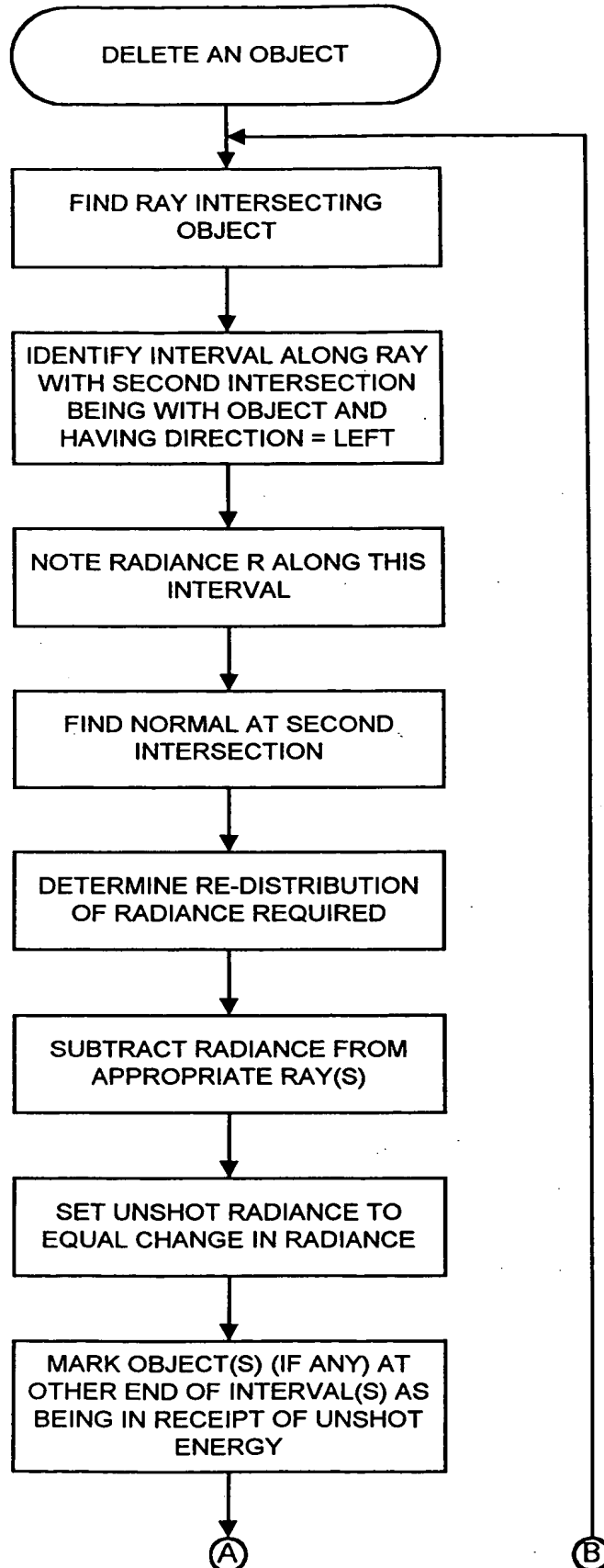


FIG. 21

This Page Blank (uspto)

26/32 .

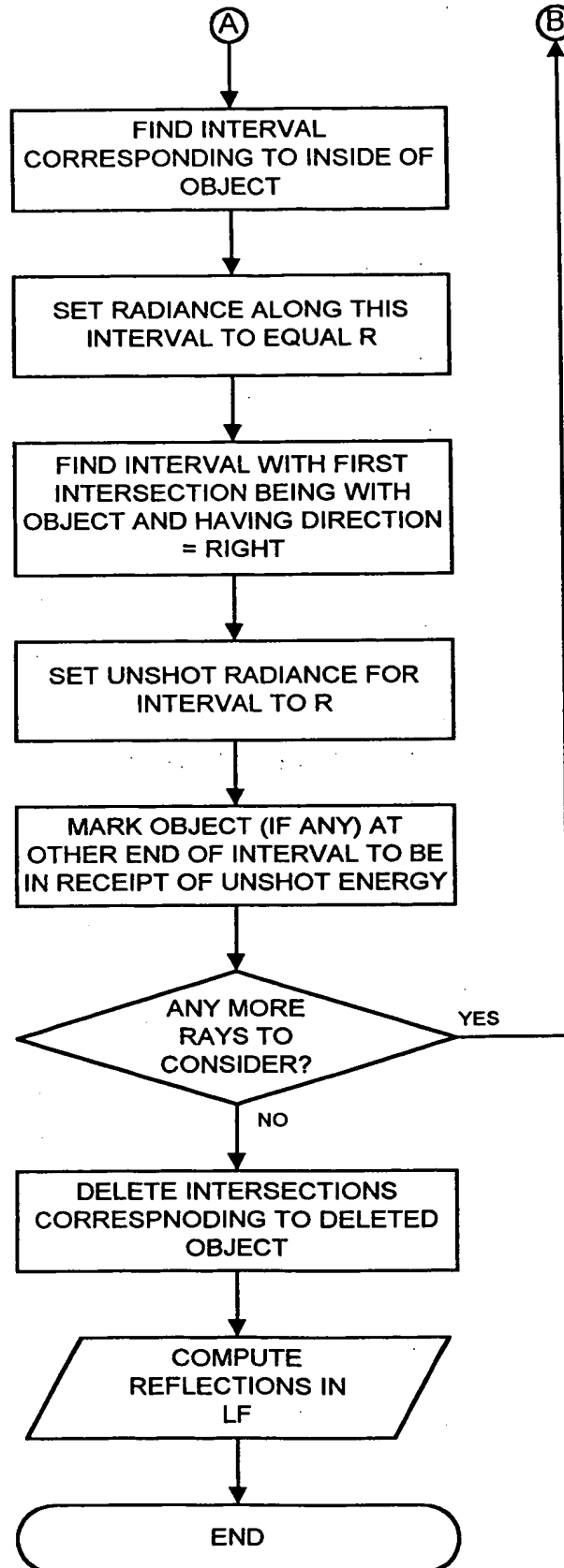


FIG. 21 (cont)

This Page Blank (uspto)

27/32

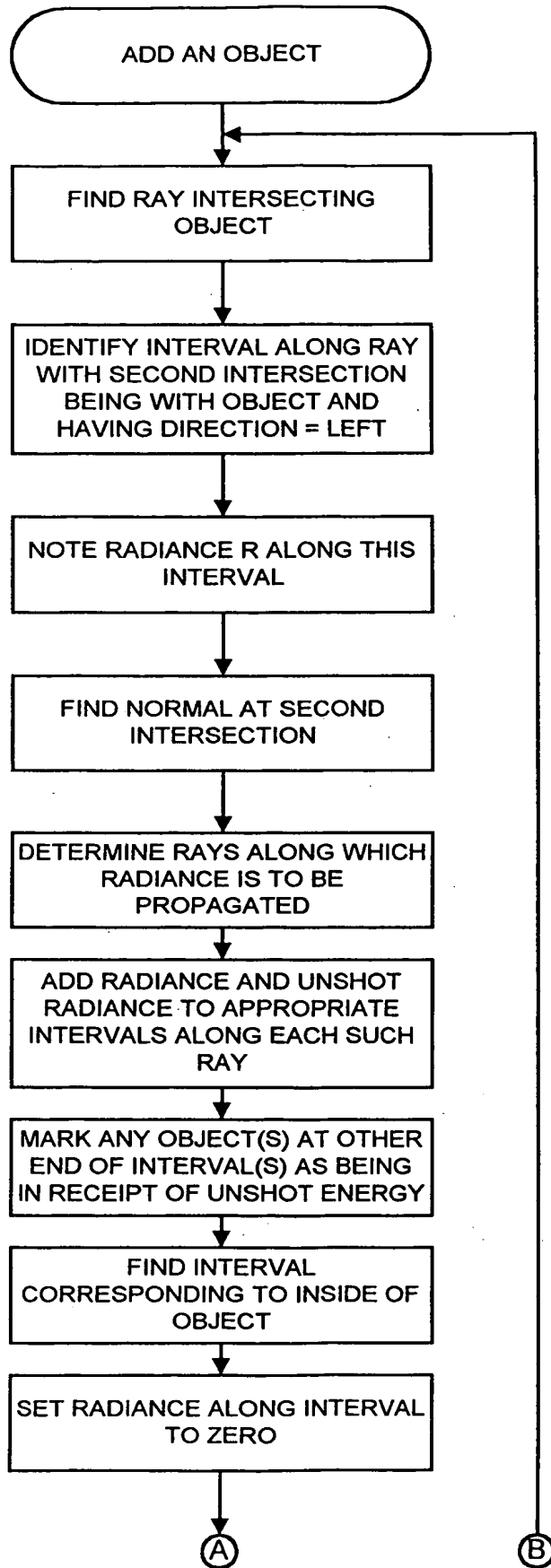


FIG. 22

This Page Blank (uspto)

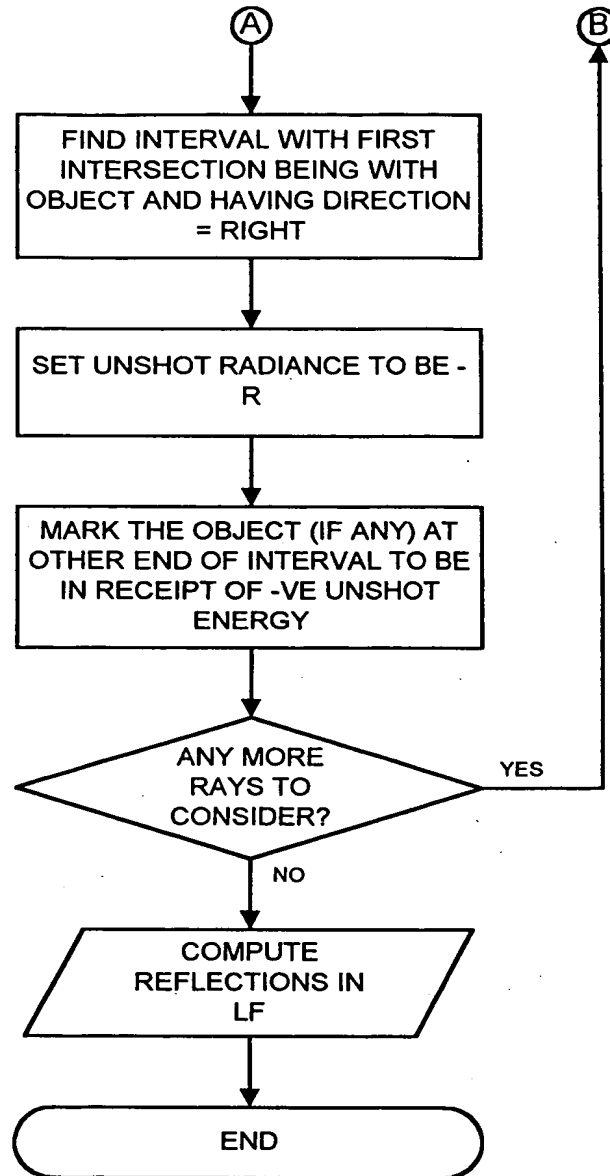


FIG. 22 (cont)

This Page Blank (uspto)

FIG. 23

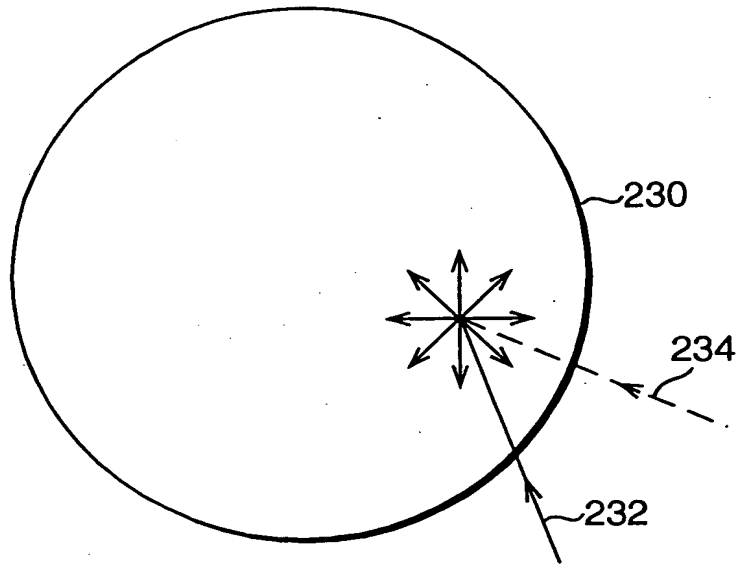
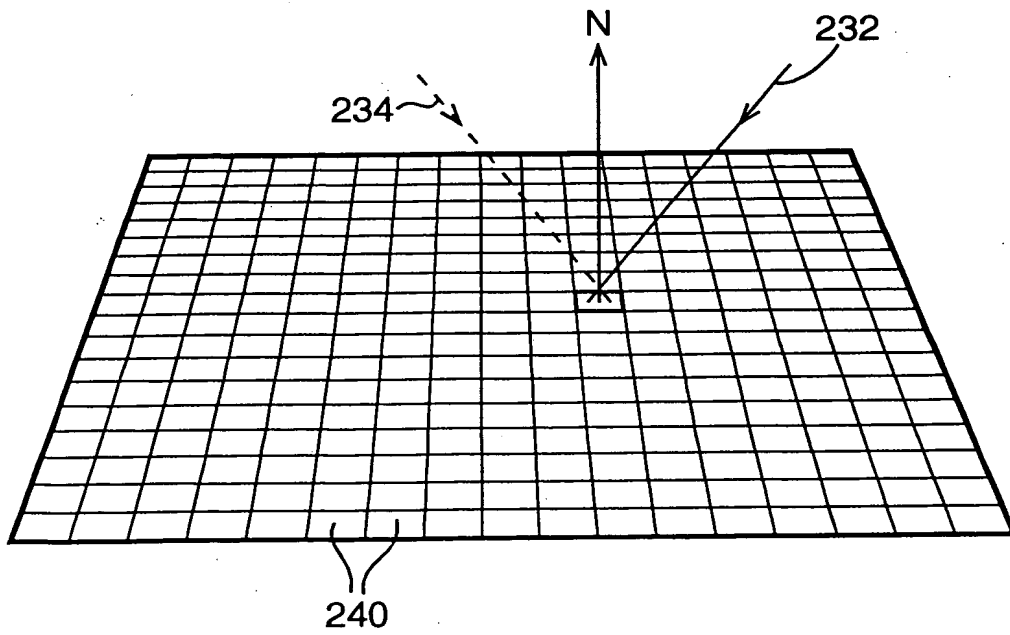


FIG. 24



This Page Blank (uspto)

FIG. 25

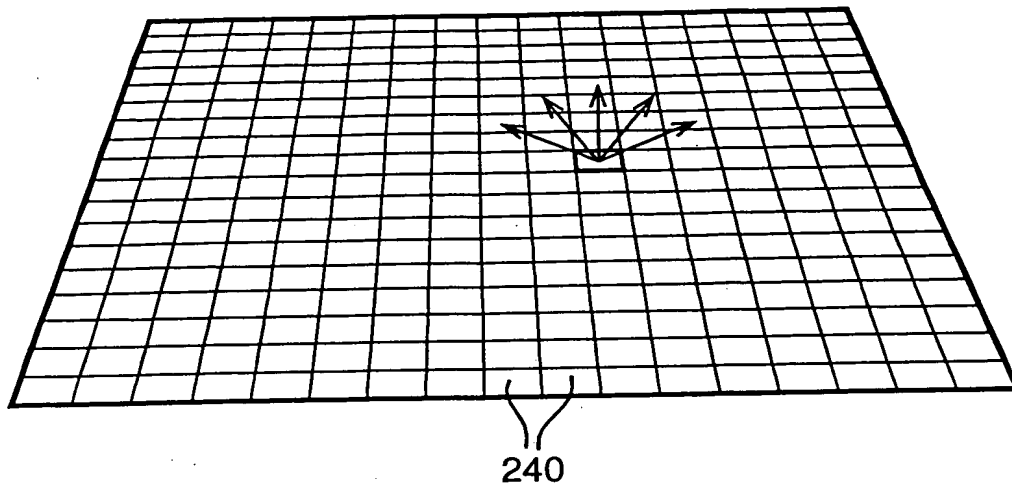
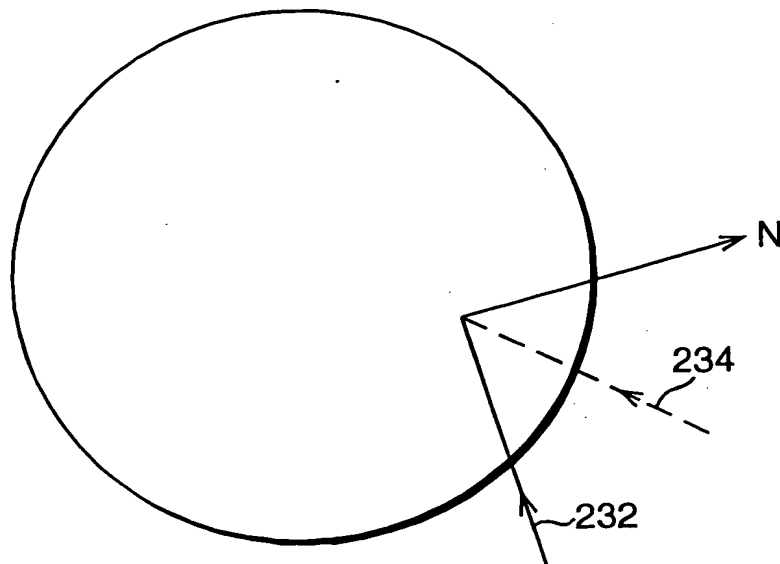


FIG. 26



This Page Blank (uspto)

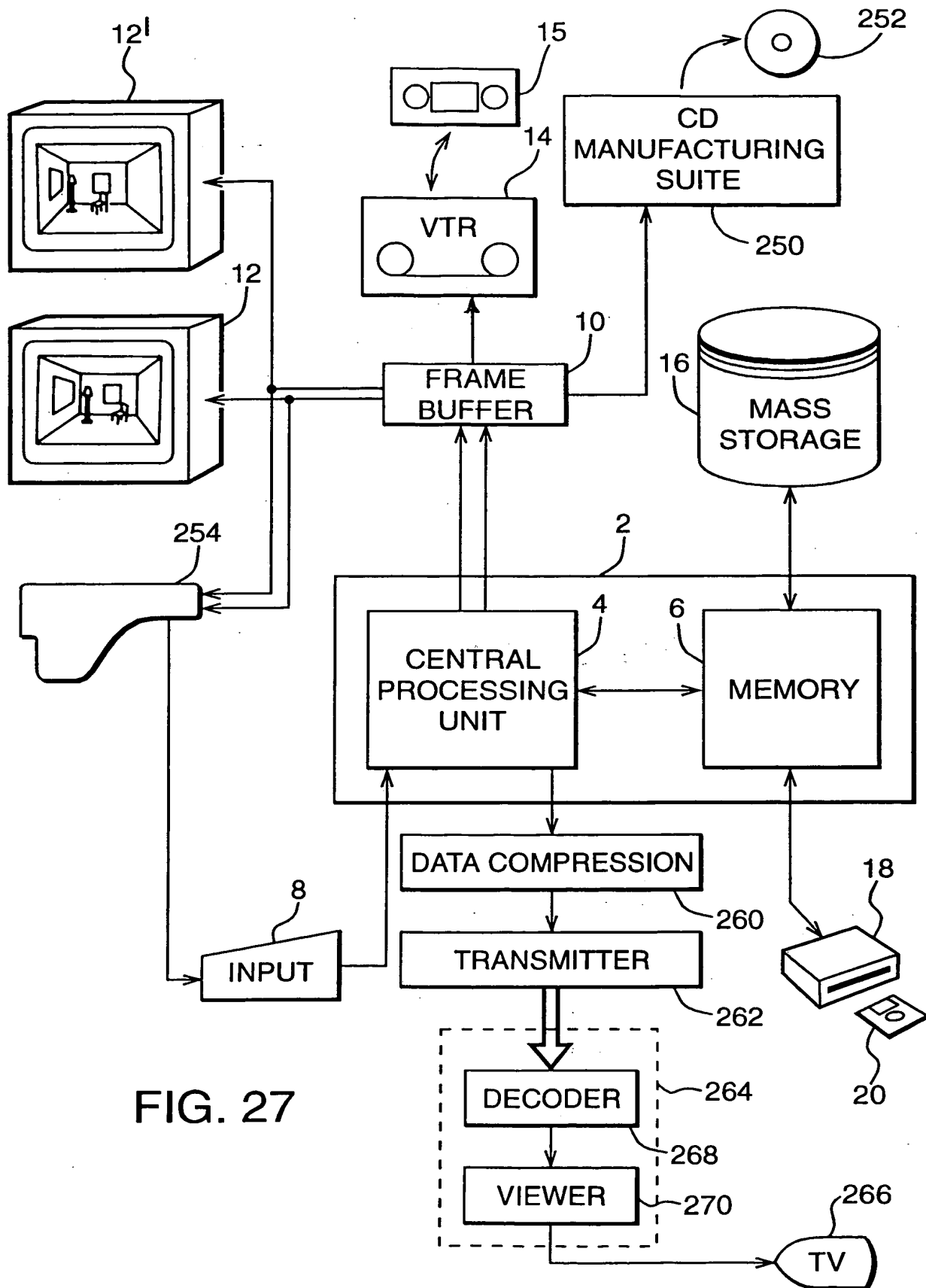


FIG. 27

This Page Blank (uspto)

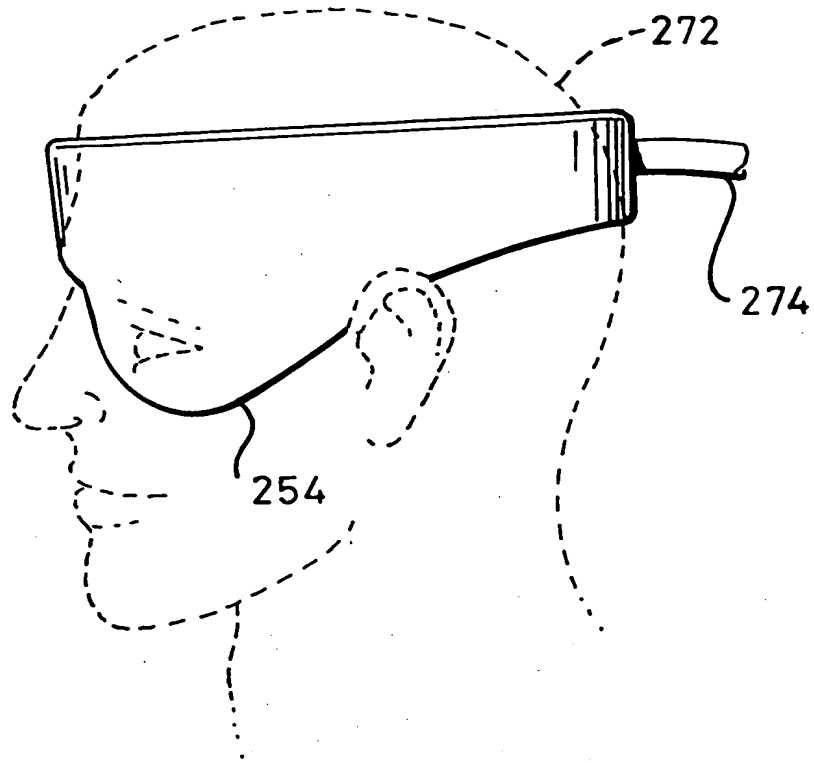


FIG. 28

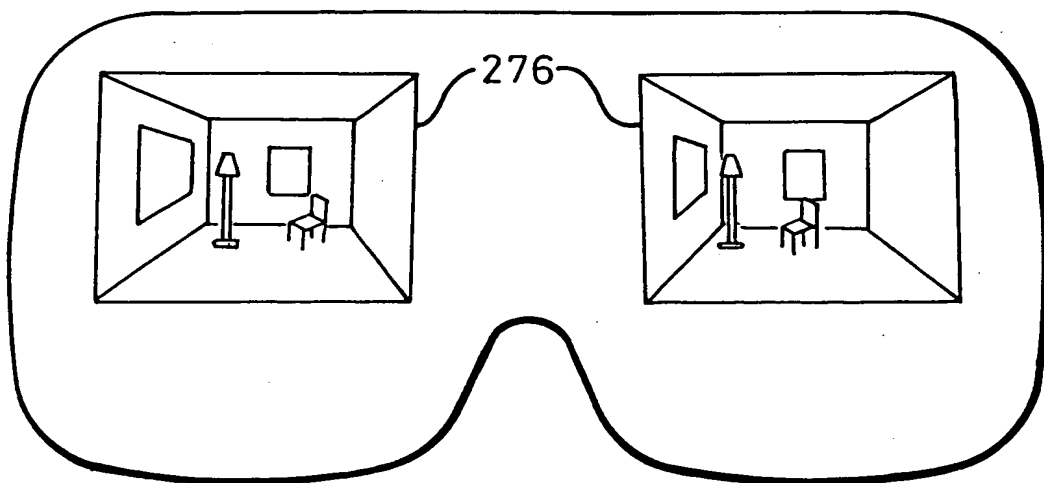


FIG. 29

ACT/SB99/03229

2/9/09

Beresford - Co.

This Page Blank (uspto)